# Tasktop Integration Hub Home

## Tasktop: 21.3 Release

July 20, 2021

# What's New

## In this release

Migrating Configuration Changes Between Environments | Protection Against Connection Lockouts and Errors | Faster State Transition Configuration | Container Mirroring Improvements | Flow Fabric™ Enhancements

## Migrating Configuration Changes Between Environments

Tasktop Hub administrators typically maintain at least two instances — one for a test environment and the other for production. With our new configuration migration feature, keeping those two instances aligned has never been easier.

Now you can export configuration changes made in one instance and migrate them to another. For example, you can replicate changes made to a test environment to your production environment.

Navigate to **History** and click **Export Changes** to export a selection or all of your changes, and then import the configuration file to your target instance. Read the documentation here.

## Protection Against Connection Lockouts and Errors

Good news for administrators dealing with infrastructure changes! If one of your tools (repositories) is temporarily unavailable (periodic maintenance, password, or URL changes), you can now also temporarily disable Hub's connection to the tool. Any integrations using this repository connection will remain disabled until you turn it back on, which will allow you to make any required configuration changes without generating errors or getting locked out. When you're ready, you can enable the repository once more and resume all its integrations. The new option is available on the **Repository** screen. Read the documentation here.

Also new in 21.3 is a database backoff configuration. By default, Tasktop will wait one hour following a database connection failure (e.g., invalid username or password) before retrying the connection. This feature is especially useful for databases with a lockout or brute force policy configured. Read more in our docs.

## Faster State Transition Configuration

Love the ability to graphically define your State Transitions? Now there's even more to love.

First, you can easily copy and reuse a state transition in a different Collection. And second, you can automatically generate an all-to-all state transition graph and edit the necessary transitions and add parameters where needed.

# Container Mirroring Improvements

Hub conveniently allows you to set up a single integration to synchronize both containers (nested folder structures) and work items within them. But often, the hierarchical container structures in the source and target repositories are different.

Now, we've added the ability to synchronize two types of containers in the source repository to one type of container on the target repository, with conditions governing container creation.

This is good news for Jama users, who can now synchronize Components, Sets, and Requirements to tools with their own nested container structure, like ALM and qTest.

A new Shared Container Mirroring sash is available for defining the conditions to create corresponding container types in the target Collection.

# Flow Fabric™ Enhancements

- For users of BMC Remedy, OAuth 2.0 is now supported via Remedy SSO for strong authentication.
- For Jama users, Hub now supports synchronizing test steps as artifacts.

# User Guide

# Welcome to User Guide

21.3 Release (July 20, 2021)

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

| Visibility | | | |
|---|---|---|---|
| Enterprise Data Stream<br><br>(Enterprise Data Stream Template) | ✗ | ✓ | ✓ |
| Integration Landscape View | ✓ | ✓ | ✓ |
| Associated Configuration Elements | ✓ | ✓ | ✓ |
| Configuration History | ✓ | ✓ | ✓ |

# Key Concepts

## Overview

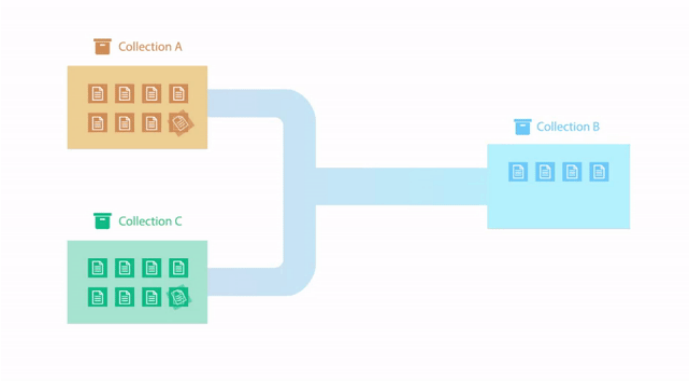
Tasktop is a powerful tool for **connecting your software delivery systems to empower teams, enhance communication, and improve the process of software development as a whole.** Below is a look at some of the concepts Tasktop utilizes to facilitate integration.

The **key concepts** to understand are:

## Integration

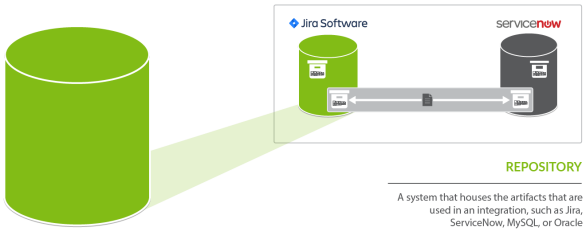
At the highest level, the definition of an **integration** is simply the flow of information between **two** or **more tools**. If you dig a little bit deeper, the definition of an integration is the flow of information, defined by the flow specification, between two or more collections. And collections 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 **R epositories**. 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

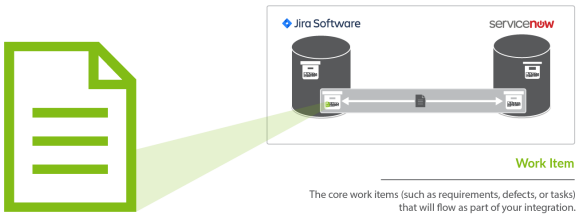
An **artifact** is **any object containing metadata that resides within your repository**. There are two main types of artifacts: **work items** and **containers**. Work items and containers have some similarities, and some key differences, with regard to how they behave within Tasktop Integration Hub.

## 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

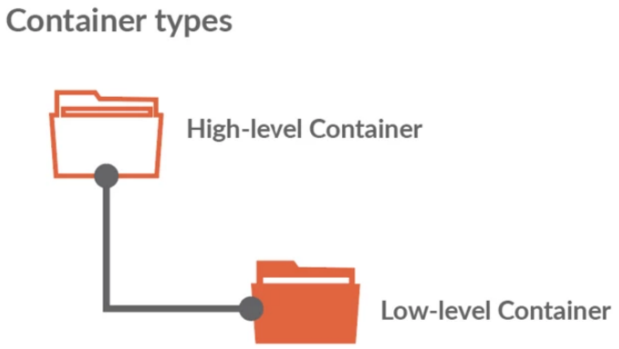
**Containers** are **artifacts that are used to group work items. They define where, within the repository, each work item resides.** Some examples of common containers are projects, folders, modules, workspaces, and sets. The main purpose of a container is to define a set of work items.

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.



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.

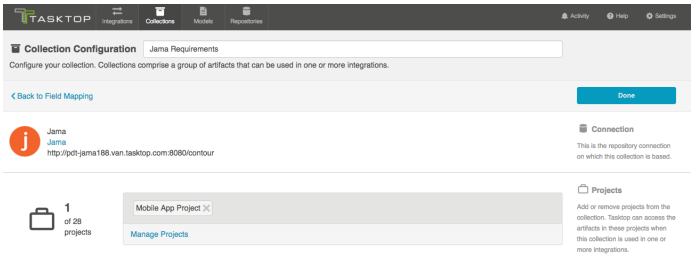
You can learn more about how to select the containers included in your collection here.

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.

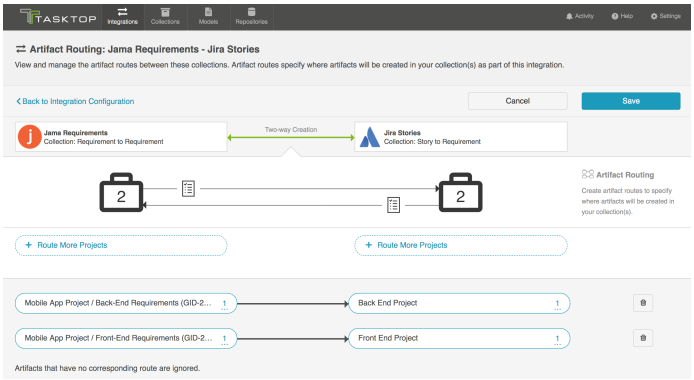
You can learn more about how to configure artifact routing here.

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.

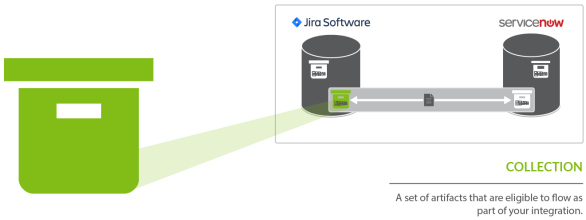
When we configure our Jama collection, we will define that collection at the *high-level container* level: this means that we can define the collection based on projects. Here, we have selected the Mobile App Project for use in our collection.




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.

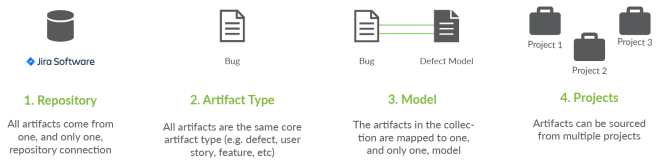



# Collection




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.)

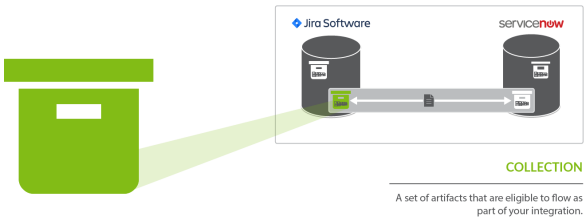
The artifacts in a collection share a set of fields that have repository-specific names and values. Part of creating a collection involves choosing a model on which to base the collection and then mapping these repository specific fields and values to those defined in the model. The concept of models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

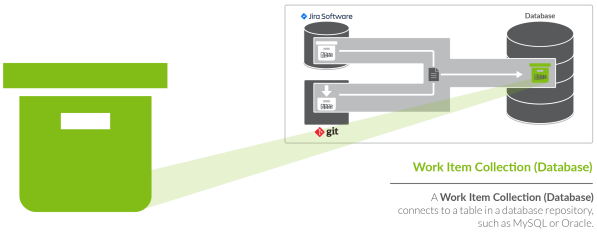



**Repository Collections (meaning either a Work Item Collection or a Container Collection that connects to a repository) comprise artifacts from an ALM, PPM, or ITSM repository like *Atlassian Jira*, *ServiceNow*, *CA Clarity*, or *Zendesk*.** When used in an integration, artifacts in a repository collection 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)



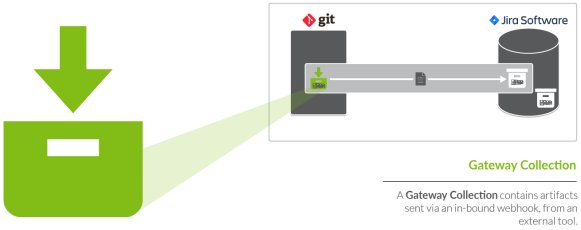

**Databases collections (a type of Work Item Collection) connect to a table in a database repository, such as MySQL or Oracle.** Artifacts in the source repository will flow data to the fields in that table.

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?

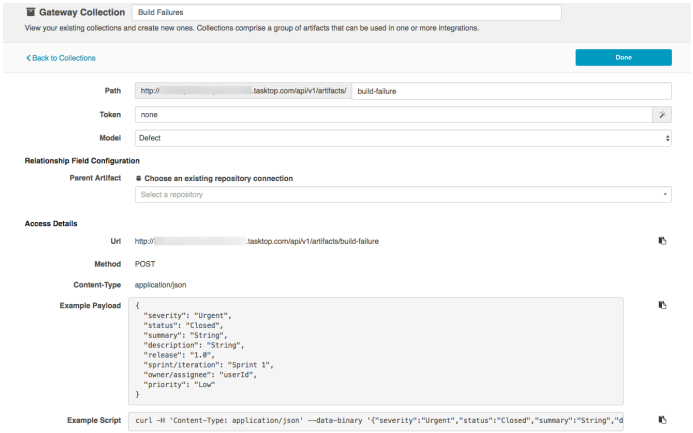
| Action | Permissible |
|---|---|
| Create artifacts in collection | ✅ |
| Update artifacts in collection | ❌ |
| Detect additions or updates to artifacts in collection in order to create or update artifacts in another collection | ❌ |

## 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:

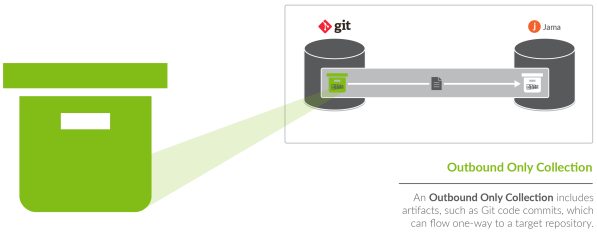



When used in an integration, artifacts in a gateway collection can trigger the creation or modification of artifacts in another collection.

### 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



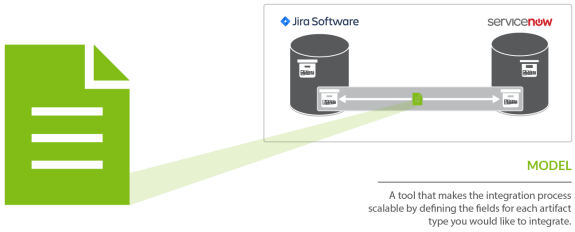

**Outbound Only Collections contain artifacts like code commits or changesets from Source Code Management repositories like *Git*, which you may want to flow out of your repository, but which would not receive updates into your repository.** When used in an integration, artifacts and information will 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

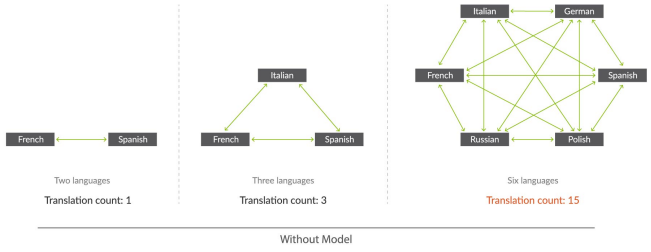
These three critical success factors are met with our usage of **models**. In very basic terms, **a model is simply a list of fields or attributes that define a certain artifact that you want to integrate.** For example, below is a very basic defect model:

| **Defect Model** | |
|---|---|
| **Field** | **Field Type** |
| Description | String |
| Priority | Single Select:<br>• High<br>• Medium<br>• Low |
| Status | Single Select:<br>• New<br>• In Progress<br>• Complete |

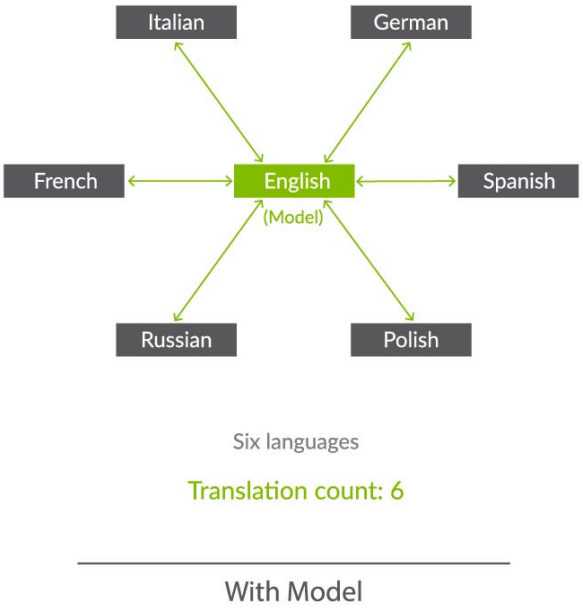
Let's talk about the first critical success factor – the ability to normalize disparate definitions of artifacts between different collections. Or, another way of thinking of it, the classic *"you say tomato, I say tomahto" conundrum*. In the diagram below it is apparent that the Jira bug is similar, 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.**

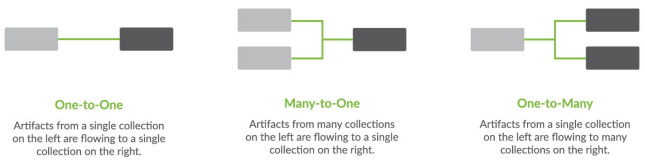
You can learn how to configure your integration using a template here.

## 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.

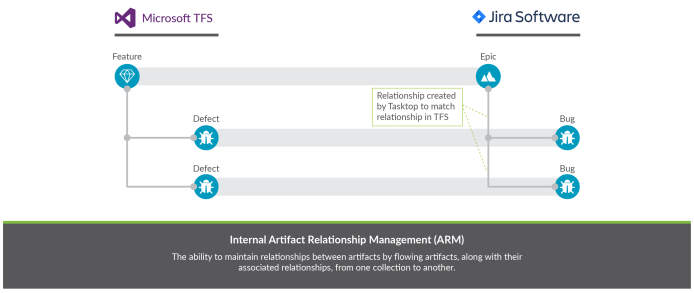
## Internal ARM

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

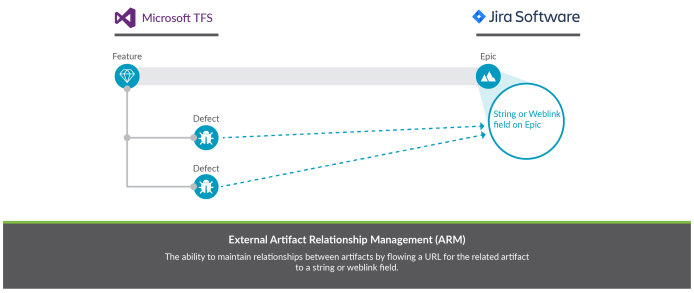



## External ARM

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 the 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:

# Installation Primer

# Overview

The Installation Primer describes how to install Tasktop Integration Hub and covers some basic information you should know before proceeding with the installation. If you are working on a deployment with Tasktop, your Solutions Architect will assist you with the installation.

On the System Requirements page, you can learn about:

- Supported Operating Systems
- Supported Browsers
- Supported Databases
- Java Runtime Environment
- Hardware Sizing for Deployment Scenarios

On the Installation page, you can learn about:

- Sandbox Environment
- Where to download Tasktop
- Installation on Windows
- Installation on Linux
- SSL Certificate Installation
- Port Configuration
- Default File Locations
- Repository Preparations

On the Advanced Configuration page, you can learn about:

- Container Configuration
- Increasing Available Memory
- Logging

On the Upgrading page, you can learn about:

- Performing Tasktop Integration Hub version upgrades
- Back up and Restore practices

On the Business Continuity page, you can learn about:

- Best practices for data loss prevention
- Impacts of Tasktop downtime
- Failover strategy/high availability guidelines

# System Requirements

⚠ Beginning in April 2022, the following operating systems will be removed from Extended Support and no longer supported by Tasktop:

- Windows Server 2012 R2
- Windows Server 2012
- Red Hat Enterprise Linux 6.x
- Ubuntu Linux 14.04 LTS
- SUSE Linux Enterprise Server 11.x

If you have any questions, please contact Support.

## General Requirements

Tasktop Integration Hub is a web application which runs centrally on a server. Users interact with it through a web browser from any computer that has network access to the server.

For best results, Tasktop Integration Hub should be deployed in an environment that has good network throughput and low latency to its operational databases and all repositories involved in an integration.

Below are general requirements to meet the needs of typical deployment scenarios.

- Tasktop Integration Hub **must** be installed in a server environment and only **one** instance of Tasktop should be installed on each server.
- The Hub operational database should have its own machine and should be co-located with the Hub server to reduce latency.

## User Requirements

To install and configure Tasktop Integration Hub, you need an account with administrative privileges on your server. The account must also have read/write access to the default file locations.

### Supported Operating Systems

⚠ **Note**: For Windows, Powershell 4 must be installed on your server.

The following 64-bit operating systems and versions are supported:

- Windows 10
- Windows Server 2019
- Oracle Enterprise Linux 7+
- Oracle Enterprise Linux 8+

- Red Hat Enterprise Linux 8.x
- Ubuntu Linux 18.04 LTS
- Ubuntu Linux 20.04 LTS
- SUSE Linux Enterprise Server 12.x
- SUSE Linux Enterprise Server 15.x

Available under Extended support:

- Windows Server 2016 *(End-of-Service-Life Date:* 18 Jan 2023*)*
- Red Hat Enterprise Linux 7.x *(End-of-Service-Life Date: 18 Jan 2023)*
- Ubuntu Linux 16.04 LTS *(End-of-Service-Life Date: 18 Jan 2023)*

# Supported Browsers

💡**Note**: Tasktop Integration Hub runs with a minimum screen resolution of **1280 pixels x 800 pixels**.

The Tasktop Integration Hub web interface is supported on the following browsers:

- Firefox 89.0+
- Google Chrome 91.0+
- Microsoft Edge Chromium 91.0+

Available under Extended Support:

- N/A

💡 If you are interested in extended support, please reach out to your Tasktop contact.

## Supported Databases for storing Tasktop Operational Data

***This feature is not applicable to Tasktop Cloud.***

⚠️ Tasktop automatically stores operational data to a pre-configured Derby database. This is suitable for evaluation purposes *only*, and is not supported for production environments. Configuring Tasktop to utilize an external database enables you to perform frequent back-ups without stopping Tasktop Integration Hub, and ensures that your Tasktop Integration Hub practices are consistent with your existing disaster and recovery process.

To reduce latency, the Hub operational database should have its own machine and should be co-located with the Hub server.

# Minimal User Permissions

For all supported databases, the user **must** have sufficient permissions to connect, create, alter and drop tables and indexes and create temporary tables. Users must also have sufficient permissions to select, insert, update, delete, and truncate tables.

Tasktop supports this operational database policy for scenarios where your database is on **any** cloud infrastructure like AWS or Azure. You can refer to the resources below for more information on encrypting communication between Hub and Database:

- For AWS, we recommend implementing a VPC. Click here for more information.
- For Azure, we recommend a VPN gateway. Click here for more information.

💡**Note**: A separate database must be used for Tasktop Operational Data and Enterprise Data Stream integrations.

### Operational Database Recommendations

The recommendations below offer a **general guideline** only. We recommend consulting with Tasktop Support to determine the exact needs for your integration scenario, and for guidance on how to efficiently configure Hub.

You can see guidelines regarding external database sizing here.

We strongly recommend using the latest supported version of **PostgreSQL** for storing Tasktop Operational Data. At scale, Tasktop Hub performs better, more reliably, and requires fewer resources with PostgreSQL than with the other available database options.

### Supported Versions and Configuration Details

In the section below, you will find supported database versions for storing Tasktop Operational Data and configuration details for each database.

## PostgreSQL

Supported Versions

- 9.6
- 10
- 11
- 12
- 13

Extended Support

- N/A

💡 If you are interested in extended support, please reach out to your Tasktop contact.

Configuration Settings

- The database must be case-sensitive (this is the default configuration).
- The database must be configured with the **UTF8** character set.

```
CREATE DATABASE dbName
ENCODING 'UTF8'
LC_COLLATE = 'en_US.UTF-8'
LC_CTYPE = 'en_US.UTF-8'
TEMPLATE template0
```

### Necessary User Permissions

The following provides sufficient permissions for the `tasktop_hub` user on the `tasktop_hub` database, when using a public schema:

```
REVOKE ALL PRIVILEGES ON DATABASE tasktop_hub
        FROM tasktop_hub;

GRANT CONNECT, TEMP ON DATABASE tasktop_hub
        TO tasktop_hub;

GRANT CREATE ON SCHEMA public
        TO tasktop_hub;

GRANT SELECT, INSERT, UPDATE, DELETE, TRUNCATE
        ON ALL TABLES IN SCHEMA public
        TO tasktop_hub;
```

The following provides sufficient permissions for the `tasktop_hub` user on the `tasktop_hub` database, when using a custom schema:

💡 If you use a custom schema, please note that when configuring the external database connection you will need to add "`?currentSchema=tasktop`" to the database connection URL, e.g. `jdbc:postgresql://example.com:5432/dbName?currentSchema=tasktop`

```
CREATE SCHEMA TASKTOP;

REVOKE ALL PRIVILEGES ON DATABASE tasktop_hub
     FROM tasktop_hub;

GRANT CONNECT, TEMP ON DATABASE tasktop_hub
     TO tasktop_hub;

GRANT CREATE ON SCHEMA Tasktop
     TO tasktop_hub;

GRANT SELECT, INSERT, UPDATE, DELETE, TRUNCATE
     ON ALL TABLES IN SCHEMA Tasktop
     TO tasktop_hub;
```

## Microsoft SQL Server

### Supported Versions

- 2017
- 2019

### Extended Support

- N/A

### Configuration Settings

- The database must be case sensitive. We recommend Latin1_General_100_CS_AS_KS_WS.
    - This can be configured using the following command (replace **dbName** with the name of your database):

      ```
      ALTER DATABASE dbName COLLATE Latin1_General_100_CS_AS_KS_WS;
      ```

    - We recommend monitoring the size of your transaction log, as very large transaction logs can cause database connection errors.
    - We recommend using JDBC driver `mssql-jdbc-7.0.0.jre8.jar`when transferring from operational database to SQL Server.

Necessary User Permissions

- The user must be a SQL authenticated user (not a Windows authenticated user)
- Additionally, the user must have the following roles granted:
    - `db_datareader`
    - `db_datawriter`
    - `db_ddladmin`

💡**Note**: Instance and Database name options can be specified by attaching "`;instanceName=;databaseName=`" to the end of the JDBC URL in Tasktop Integration Hub.

## Oracle

Supported Versions

- 18c
- 19c

Extended Support

- N/A

💡 If you are interested in extended support, please reach out to your Tasktop contact.

Configuration Settings

- The database must be case-sensitive (this is the default configuration).
- The database must be configured with the **AL32UTF8** character set.

  ```
  ALTER DATABASE dbName CHARACTER SET AL32UTF8;
  ```

Necessary User Permissions:

User must have `CREATE SEQUENCE`, `CREATE TABLE`, `CREATE SESSION` permissions, as well as sufficient quota. Typical user creation might look as follows:

```
CREATE USER tasktop_hub IDENTIFIED BY a_password DEFAULT TABLESPACE tasktop_hub;

GRANT CREATE SESSION TO tasktop_hub;

GRANT CREATE SEQUENCE, CREATE TABLE TO tasktop_hub;
ALTER USER tasktop_hub QUOTA UNLIMITED ON tasktop_hub;
```

### Troubleshooting

- To resolve error **ORA-30036 (UNABLE TO EXTEND SEGMENT BY 8 IN UNDO TABLESPACE)**, please refer to the following documentation.

## MySQL

### Supported Versions

- 5.7.7+ (excluding 5.7.0 - 5.7.6)
- 8.0

### Extended Support

- N/A

### Configuration Settings

The following settings must be applied before connecting Tasktop to MySQL:

- The database must be case sensitive.
  - This can be configured using the following command (replace **dbName** with the name of your database):

    ```
    ALTER DATABASE dbName COLLATE = 'utf8_bin'
    ```

  - The database default charset must be UTF-8, `ALTER DATABASE dbName CHARACTER SET = 'utf8'`
    - You can also create the database with these settings: `CREATE DATABASE dbName CHARACTER SET = 'utf8'`
  - We recommend using JDBC driver version 8.0 or later when transferring from an operational database to MySQL Server.
  - `innodb_default_row_format` must be `DYNAMIC`
  - `innodb_file_format` must be `Barracuda`
  - `innodb_file_per_table` must be `ON`
  - `innodb_large_prefix` must be `ON`
  - `innodb_buffer_pool_size` must be minimum 1G
    - This size is highly dependent on customer hardware and data size — the number above is only a recommendation. Please consult with Tasktop Support if you have any questions.
  - `max_allowed_packet` property must be minimum 64M

    - If this is set too low, you will see a **Packet for query is too large** error on the Activity screen.
- `max_connections` property should be minimum 500
    - **Note**: The number of connections Tasktop uses is highly dependent on customer configuration, hardware, and load — the number above is only a recommendation. Please consult with Tasktop Support if you have any questions.

💡**Note**: `innodb` settings are the default settings for MySQL, so you will not need to make any changes to those settings unless they have been changed previously. The `innodb` settings apply globally to all MySQL databases on the server, while the `character set` is specific to the database.

⚠ Configuring Tasktop Integration Hub with the MySQL external operational database will prohibit the synchronization of 4-byte characters due to MySQL's default UTF8 encoding being limited to 3 bytes. Examples of 4-byte characters include but are not limited to some emojis and some Chinese characters. If you may be synchronizing 4-byte characters, consider using another supported database.

Necessary User Permissions

The following provides sufficient permissions for the `tasktop_hub` user on the `tasktop_hub database`:

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM tasktop_hub;

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, INDEX, LOCK TABLES, REFERENCES ON tasktop_hub.* TO
tasktop_hub
```

# Supported Databases for use in Enterprise Data Stream Integrations

The Tasktop Database add-on allows you to create integrations that send artifact information to one central database.

💡**Note**: A separate database must be used for Tasktop Operational Data and Enterprise Data Stream integrations.

You can see guidelines regarding external database sizing here.

## Supported Versions

If your license includes the Tasktop Database add-on and you would like to configure an Enterprise Data Stream Integration, the following databases and versions are supported:

### PostgreSQL

General Support

- 9.6 - 13

Extended Support

- N/A

💡 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

- 18c
- 19c

Extended Support

- N/A

## 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

For PostgreSQL, please refer to PostgreSQL documentation for more information.

### Location

- Example Format: *jdbc:postgresql://hostServerName:postgreSqlServerPort/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;databasename=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**)

💡 **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**).

# Java Runtime Environment

Tasktop Integration Hub is packaged with a JRE; there is no need to install a JRE separately. Tasktop Integration Hub uses and ships with Oracle Java.

💡 **Note**: Partner branded editions of Tasktop Integration Hub use and ship with **Azul OpenJDK**.

# Deploying Hub on a Cloud Environment

⚠️ To ensure reliable performance, all virtual machines (on-prem and private cloud) **must** meet the requirements listed in the General Requirements section.

Tasktop Hub can be deployed in operating systems on physical servers within virtual machines hosted on dedicated on-prem virtual machine hosts. Tasktop Hub can also be deployed within private cloud deployments, such as AWS or Azure. If deploying on a private cloud environment, Hub and its operational database must be deployed using a full image and not a container, with the exception of AWS RDS PostgreSQL. Tasktop Hub **cannot** run in containerized deployments (Kubernetes, OpenShift, etc.).

Tasktop has qualified AWS RDS PostgreSQL deployments for use as the operational database for Hub instances hosted in AWS private cloud environments. Tasktop does not offer direct support for private cloud hosting infrastructure (i.e., AWS networking and configuration) beyond the operation of Tasktop's own products within the hosted environment. See the section below for recommended configuration settings.

## AWS RDS PostgreSQL Recommendations

The recommendations below offer a **general guideline** only. We recommend consulting with Tasktop Support to determine the exact needs for your integration scenario, and for guidance on how to efficiently configure Hub.

**Note**: Tasktop does not troubleshoot or maintain AWS RDS PostgreSQL. Please ensure your database is configured in a way that is aligned with your security policy and that the associated risks are understood and accepted.

| Setting | Recommended Value |
|---|---|
| `"DBInstanceClass"` | `"db.t3.small"` |
| `"Engine"` | `"postgres"` |
| `"AllocatedStorage"` | `50` |
| `"BackupRetentionPeriod"` | `30` |
| `"MultiAZ"` | `true` |
| `"EngineVersion"` | `"13.6"` |
| | |

| | |
|---|---|
| "AutoMinorVersionUpgrade" | false |
| "PubliclyAccessible" | false |
| "StorageType" | "gp2" |
| "StorageEncrypted" | true |
| "CopyTagsToSnapshot" | true |

# Hardware Sizing for Deployment Scenarios

## General Notes and Considerations

Below are recommendations on sizing hardware and virtual machine capacity to meet the needs of typical deployment scenarios.

These recommendations are guidelines intended to provide a starting point when deciding on hardware allocation for a specific deployment. We recommend monitoring system load including CPU usage, memory pressure and disk queue length, and adjusting the system sizing accordingly.

For best results, Tasktop Integration Hub should be deployed in an environment that has good network throughput and low latency to all repositories and databases involved in an integration.

Based on real-life metrics, we approximate database sizing at about 40 KB per artifact. For 100,000 artifacts total (including artifacts on both sides of an integration), that equates to about 4 GB of database storage, not including log files, rollback space, etc.

This is a rough estimate, and will depend on customer-specific configuration and usage. For example, artifacts that have hundreds of fields and many large comments will require more space. Likewise, short change detection intervals, frequent full scans, or frequent changes to large numbers of artifacts will require more processing power.

## Hub Server Sizing Recommendations

⚠ The recommendations below offer a **general guideline** only. The performance needs of Hub integrations depend on how integrations are configured, the specifications of connected end systems, and the volume and type of changes made in the end systems.

Note that it is possible for a deployment to have a low number of integrations and users, but a high number of artifact updates (or vice versa). We recommend consulting with Tasktop Support to determine the exact sizing needs for your integration scenario, and for guidance on how to efficiently configure Hub.

## Small Deployment

A deployment managing up to 20,000 artifacts in up to 100 projects with up to 10,000 updates/month (typically up to 200 active users, and up to 5 integrations).

- 4 GB system memory
- 3 GHz processor, 2 cores
- 50 GB free disk space

## Medium Deployment

A deployment managing up to 150,000 artifacts in up to 500 projects with up to 50,000 updates /month (typically up to 1,000 active users, and up to 15 integrations).

- 8 GB system memory
- 2 x 3 GHz processor, 4 cores
- 150 GB free disk space

## Large Deployment

A deployment managing up to 1,000,000 artifacts in up to 2000 projects with up to 200,000 updates /month (typically more than 2,000 active users, and 20+ integrations).

- 16 GB system memory
- 4 x 3 GHz processors, 8 cores
- 250 GB free disk space

## Extra-Large Deployment

If your deployment exceeds any of the guidelines from the **Large Deployment**, please consult with Taskt op Support.

For extra-large deployments, the specific characteristics of the integrations are crucial when determining proper instance sizing. As a result, no general recommendations can be offered for extra-large deployments.

# External Database Sizing

The system that the external database is deployed on should also follow the sizing recommendations listed above. For example, the database for a large deployment should run on a separate machine with 16 GB of memory, 8 cores, and 250 GB of disk space.

# Java Heap Size

We recommend setting the maximum Java heap size value to 50-75% of your system's memory.

Learn more about setting Java heap size here.

# Installation

## Sandbox Environment

It is recommended that you prepare a sandbox environment to test your Tasktop Integration Hub configuration before deploying it in production.

The sandbox environment should include the following:

- A sandbox server to install Tasktop Integration Hub on
- Sandbox instances of all repositories you will be integrating
  - These instances should include the same project structure and customizations as your production repositories.
  - These instance should also include a comparable number of artifacts to your production repositories.

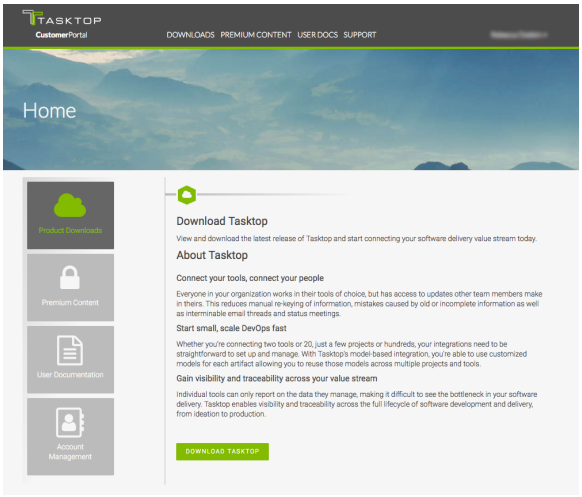
After you have configured Tasktop Integration Hub on the sandbox server and are satisfied with the way it is running with your sandbox repositories, you can install Tasktop Integration Hub on your production server and recreate the configuration for your production repositories.

## Installation

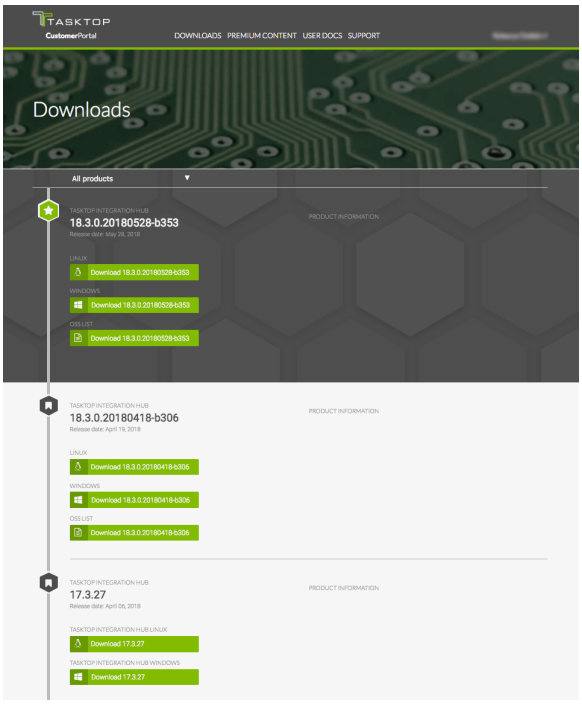
### Where to Download Tasktop Integration Hub

To get the latest version of Tasktop Integration Hub, create an account on our Customer Portal, then contact your Solutions Architect or Tasktop Support.

Once logged in to the Customer Portal, click **Product Downloads**.




This will lead you to the **Downloads** section, where you can download the latest version of Tasktop Integration Hub.

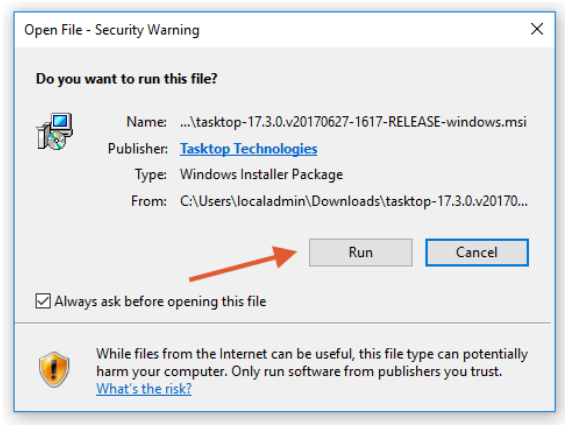



# Installation on Windows

Click the **Windows** download link on the **Product Downloads** page of the Customer Portal.

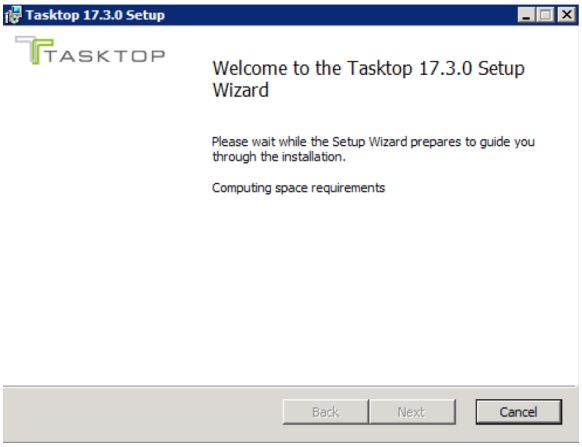
You will be provided with an installation package for Tasktop Integration Hub as a standard Windows MSI installer.

If prompted, click **Save File** and open the file once downloaded.



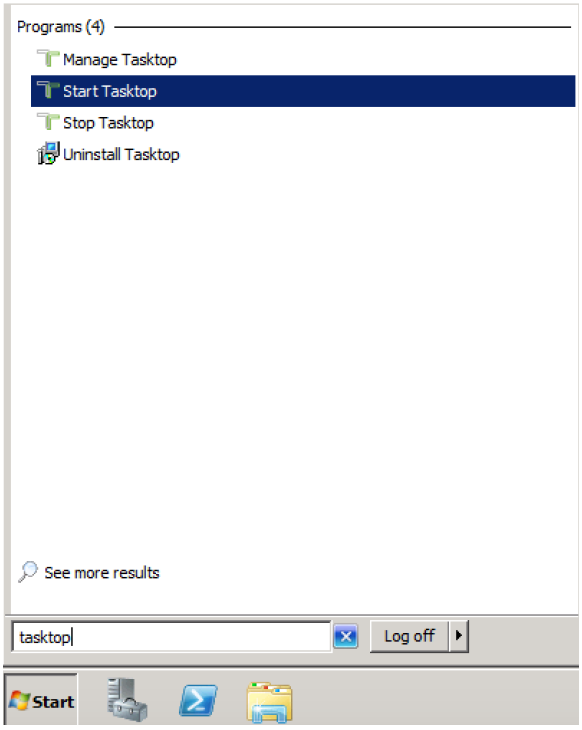

The Tasktop Setup Wizard will guide you through the installation process.

⚠ **Note**: If you decide to change the location of the ProgramData directory, do **not** include spaces in the new directory name. If the directory includes spaces, Tasktop's UI will **not** be accessible.




After installing Tasktop, open the **Start** menu and click **Start Tasktop** to start both Tasktop and User Management services.

To stop both Tasktop and User Management services, click **Stop Tasktop**.




⚠ **Note**: The Tasktop application is available via HTTPS on port 8443. A default SSL certificate is provided for testing purposes, however this SSL certificate is insecure. **Before using in a production environment, the provided SSL certificate must be replaced**. Please see details in the SSL Certificate Installation section below.

💡 Please follow the steps in the Getting Started section when starting Tasktop Integration Hub for the first time.

# Installation on Linux

## For Direct Customers

Click the **Linux** download link on the **Product Downloads** page of the Customer Portal.

You will be provided with an installation package for Tasktop Integration Hub as a `.tar.gz` archive.

To extract this archive to your desired location, copy the archive to the correct location on your Linux system.

You must choose a location with **no spaces** in its path and use the following command to extract:

```
$ tar -xzvf tasktop-linux-x64-<version>.tar.gz
```

After extracting, run the `start-tasktop.sh` script from the installation directory (see note on permissions below) to start Tasktop and User Management services.

To stop Tasktop and User Management services, use the `stop-tasktop.sh` script in the same folder.

💡 Please follow the steps in the Getting Started section when starting Tasktop Integration Hub for the first time.

## For OEM Customers

You will be provided with an installation package for Tasktop Integration Hub with no file extension in the name.

To execute the file, run these commands:

```
chmod +x tasktop-linux-x64-<version>
```

```
./tasktop-linux-x64-<version>
```

After approving the End User License Agreement, the file will automatically unzip, allowing you to run Tasktop Integration Hub.

Run the `start-tasktop.sh` script from the installation directory (see note on permissions below) to start Tasktop and Keycloak User Management services.

To stop Tasktop and User Management services, use the `stop-tasktop.sh` script in the same folder.

⚠️ The Tasktop application is available via HTTPS on port 8443. A default SSL certificate is provided for testing purposes, however this SSL certificate is insecure. **Before use in a production environment, the provided SSL certificate must be replaced.** Please see details in the SSL Certificate Installation section b elow.

💡 Please follow the steps in the Getting Started section when starting Tasktop Integration Hub for the first time.

### Note on Permissions

We recommend creating a dedicated user for running Tasktop Integration Hub. We do **not** recommend running Tasktop Integration Hub as root, as it may create files that cannot be accessed when running Tasktop as another user. Running an application on a Linux system as root may also interfere with your system's security.

For this reason, `start-tasktop.sh` will not start if it detects the current user is root.

If you would like to run Tasktop Integration Hub as root despite these risks, you can do so by deleting or commenting lines 3-7 of `start-tasktop.sh` as shown below:

```
#!/bin/sh
#if [ "`id -u`" -eq "0" ]
#then
#    echo "Tasktop should not be run as root"
#    exit 1
#fi
currentdir="$( cd "$(dirname "$0")" ; pwd -P )"
keycloak_running() {
  pgrep -n -f "${currentdir}"/keycloak/bin/standalone.sh
}
```

## Tasktop Integration Hub Service on Linux

There are several ways to configure a Tasktop Service that starts automatically on system startup. We recommend using a dedicated account for running Tasktop Integration Hub.

You can see the examples below for **SysVinit** and **Systemd**.

Tasktop Integration Hub Service with Systemd

1. Navigate to `/etc/systemd/system`
2. Create a file named `tasktop.service`
3. Paste the following into the file:

```
# Systemd unit file for tasktop
[Unit]
Description=Tasktop Integration Hub
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/path/to/tasktop/start-tasktop.sh
ExecStop=/path/to/tasktop/stop-tasktop.sh

User=user
Group=group

[Install]
WantedBy=multi-user.target
```

a. Change both instances of `/path/to/tasktop` to the full path to your Tasktop Integration Hub installation directory
b. Change the User and Group variables to the username and group of the account you want to run the Tasktop Integration Hub service

4. Reload Systemd

```
$ systemctl daemon-reload
```

5. Enable the new Tasktop Integration Hub service to start on system startup

```
$ systemctl enable tasktop
```

To manually start and stop the Tasktop Integration Hub Service, use the following commands:

```
$ systemctl start tasktop
$ systemctl stop tasktop
```

#### Tasktop Integration Hub Service with SysVinit

1. Navigate to `/etc/init.d`
2. Create a file named `tasktop`
3. Paste the following into the file:

```
#!/bin/bash
# description: Tasktop Start Stop Restart
# processname: tasktop
# chkconfig: 2345 20 80
TASKTOP_HOME=/path/to/tasktop
case $1 in
start)
sh $TASKTOP_HOME/start-tasktop.sh
;;
stop)
sh $TASKTOP_HOME/stop-tasktop.sh
;;
restart)
sh $TASKTOP_HOME/stop-tasktop.sh
sh $TASKTOP_HOME/start-tasktop.sh
;;
esac
exit 0
```

a. Change the TASKTOP_HOME variable to the full path to your Tasktop Integration Hub installation directory
b. If you'd like, you can change the `chkconfig` run levels and start and stop priorities

4. Set the permissions of Tasktop to make it executable:

```
$ chmod 755 tasktop
```

5. Use the `chkconfig` utility to enable Tasktop Integration Hub start at system startup

```
$ chkconfig --add tasktop
$ chkconfig --level 2345 tasktop on
```

   a. If you'd like, you can change the run levels in this command

To manually start and stop the Tasktop Integration Hub Service, use the following commands:

```
$ service tasktop start
$ service tasktop stop
$ service tasktop restart
```

# SSL Certificate Installation

⚠The Tasktop application is available via HTTPS on port 8443. **A default SSL certificate is provided for testing purposes and should be replaced after installation.**

Replacing the default SSL certificate used by Tasktop Integration Hub involves the following:

1. Preparing a Java keystore file with all keys and certificates
    a. The Tasktop and Keycloak SSL configuration require a JKS format keystore.
        i. If your corporate CA provides a JKS keystore file, you can skip to the **Configure Tasktop to use the keystore** section and follow the steps using the JKS keystore file from your CA.
        ii. If your CA requires you to provide a CSR and returns a certificate response to you, use the following steps to generate your own keystore file and CSR:
            1. Create a Java keystore file and generate a new key pair
            2. Generate a certificate request file
            3. Submit the file to a Certificate Authority (CA) and obtain the certificate and CA certificate trust chain
            4. Import the certificates to the keystore file
2. Configuring Tasktop to use the keystore (i.e., new key and certificate)

The SSL certificate should contain DNS names where the Tasktop server is accessible. The user's browser will verify that the name in the address bar matches the names listed in the certificate. Certificate Authority may be your internal corporate service, or you may use a public CA (e.g., Comodo, Let's Encrypt). If you are planning to use a certificate from a public CA, your Tasktop instance must have a publicly recognizable DNS name that is owned by your organization.

SSL-related instructions on this page are provided as a **reference only**. Your Certificate Authority will have more detailed instructions on creating and importing certificates. These instructions are based on the use of a GUI tool Portecle, which can be downloaded here.

Note that Tasktop does not provide support for this third-party tool beyond the instructions shown below.

💡 **Tip**: You can create the Java keystore file on any machine and move the file to the server running Tasktop software; there is no need to install Portecle on the server running Tasktop.

If you cannot use Portecle and need to utilize standard Java command line utility keytool, please refer to Tomcat documentation. Upon following the documentation, use JRE installed with Tasktop software in the Tasktop installation directory (default `C:\Program Files\Tasktop`). Tasktop's `server.xml` file is located in Tasktop's data directory (default: `C:\ProgramData\Tasktop`, or the location where Tasktop is installed on Linux) under `container/conf/server.xml`.

## Running Portecle for SSL certificate installation

To run Portecle for SSL certificate installation, see the instructions below:

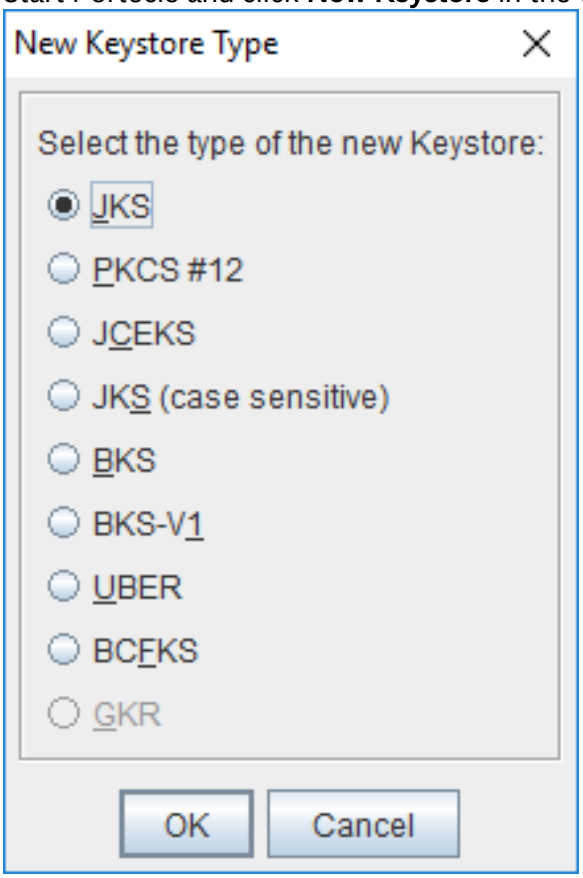
1. Download and unzip Portecle.
2. Open the command prompt.
    a. For **Windows**, navigate to `C:\Program Files\Tasktop\jre\bin\`
    b. For **Linux**, navigate to `<tasktop-install>/jre/bin/`
3. Run the following command (changing `/path/to/portecle/` to the location where you unzipped Portecle):
    a. 
```
java -jar /path/to/portecle/portecle.jar
```

## Prepare a Java keystore file with all the keys and certificates

To replace Tasktop's default SSL certificate using Portecle, follow the instructions below:

**Tip**: Details on accessing Portecle can be found in the section above.

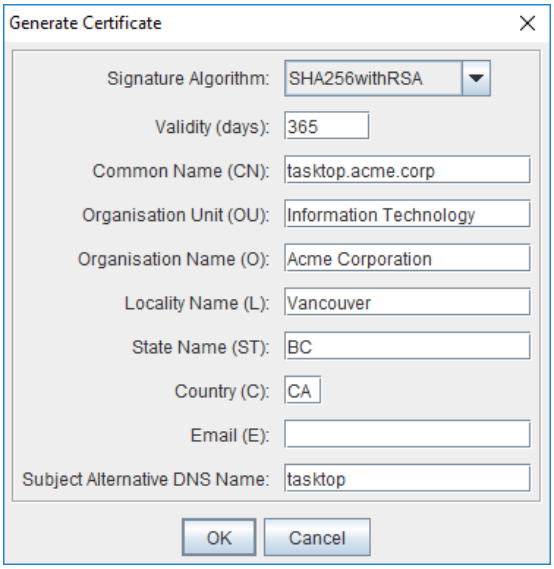
1. Create a key pair and keystore:
    a. Start Portecle and click **New Keystore** in the toolbar and select **JKS** as the keystore type.




    b. Click **Generate Key Pair** in the toolbar. You can leave the default settings for 2118 bit RSA key, or choose different settings if required by your company's security policy.

    c. In the **Generate Certificate** pop-up, enter the Fully Qualified Domain Name (FQDN) of your Tasktop server in the Common Name (CN) field and enter other fields as needed.
        a. In the **Subject Alternative DNS Name** field, enter the alternative domain name of the server, if one exists. Your certificate should include all DNS names that your users

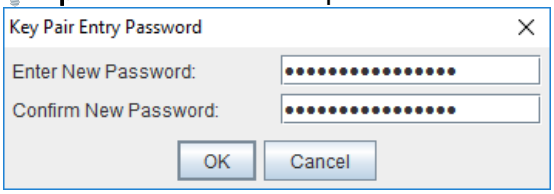
may use to connect to Tasktop. For internal corporate CA you can also use “short” names (i.e., tasktop, in addition to tasktop.acme.corp). In CA, these additional DNS names are called Subject Alternative Names, or SAN. You can specify one SAN at this point, and can usually add more names later when submitting your request to the CA.



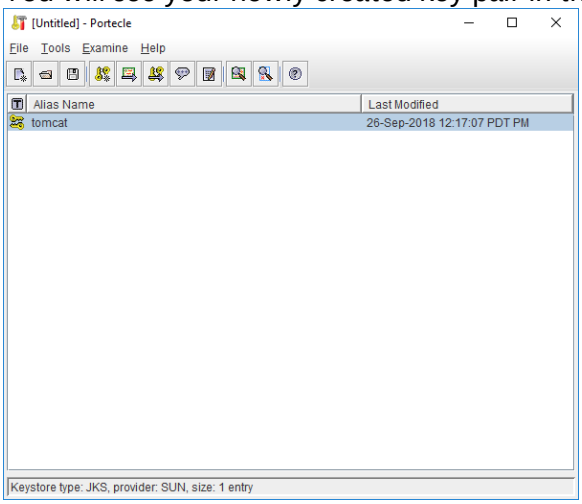

d. Enter **tomcat** as alias.

e. Create a new password for the key pair.

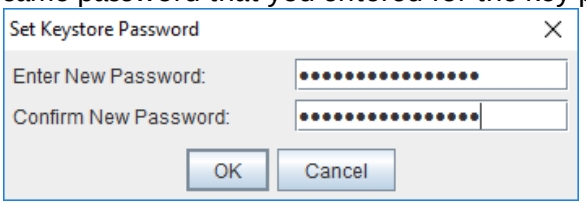
   a. **Tip**: You will need this password later when configuring Tomcat.



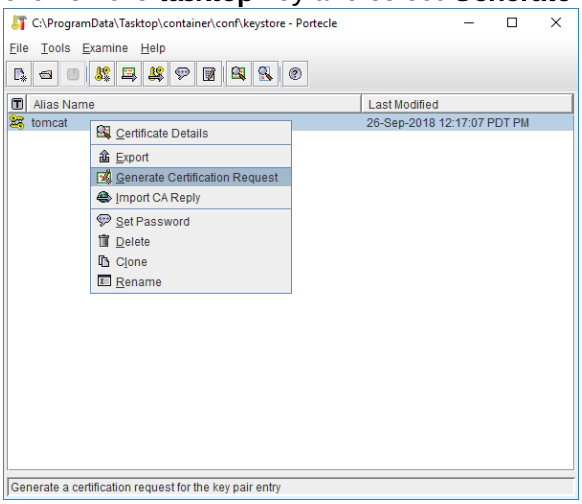

f. You will see your newly created key pair in the list.



g. Click **Save Keystore** in the toolbar to save the newly created keystore file. Here, use the same password that you entered for the key pair earlier.



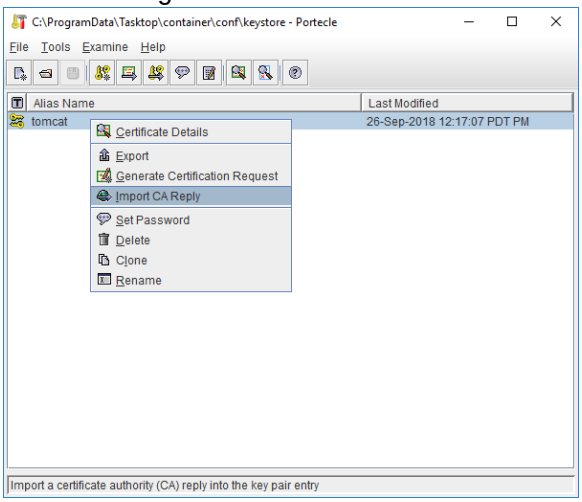

2. To generate a certificate request file (also known as Certificate Signing Request or CSR), right click on the **tasktop** key and select **Generate Certification Request** and save it to a file.




3. Submit your CSR to a CA to obtain a Certificate.

**Note**: For some CAs you will need to provide the list of all DNS names for your Tasktop server separately as they will ignore the SAN values in the certificate request. See your CA's documentation for more information.

4. Import the certificates to the keystore file.
   a. If your CA provided a separate file with the CA certificate or trust chain, import it by selecting **Import Trusted Certificate** in the toolbar. If your CA provided only one file in response to your CSR, skip to 4b.
   b. Import the server certificate by right clicking on the **tasktop** key, selecting **Import CA Reply** and choosing the server certificate file received from the CA.




   c. To verify the certificate chain, click **Tools** and then click **Keystore Report**.

## Configure Tasktop to use the keystore

1. Place your keystore file in a protected location that will not be wiped on Tasktop upgrade. We suggest using Tasktop data directory (default `C:\ProgramData\Tasktop`, or the home directory of the user that Tasktop service is running as on Linux).
2. Open the `tasktop-hub.properties` file and configure the following properties:
    a. `server.ssl.key-store` - Location where the keystore file exists
    b. `server.ssl.key-store-password` - Password of keystore file
    c. `server.ssl.key-store-type` - Type of keystore file (e.g., JKS, PKCS12)
3. Restart Tasktop Integration Hub Service

💡 To learn more about creating a `tasktop-hub.properties` file, please see the section below.

By default, the SSL configuration has been configured to disable known weak ciphers. As new security information becomes available, the list of enabled ciphers should be updated accordingly.

### Configure Keycloak User Management to use and trust Tasktop's keystore

In Tasktop Integration Hub 20.4, both Tomcat and Jboss share the same properties in the `tasktop-hub.properties` file as they share the same keystore file. See more details above.

💡 To learn more about creating a `tasktop-hub.properties` file, please see the section below.

# Port Configuration

By default, Tasktop utilizes the ports listed in the table below.

If any of those ports are already being utilized for other purposes, you will need to change them. To view a list of all ports being used on your system, you can use the netstat-a command. This will help you determine which available ports you would like to use for Tasktop.

Here is a summary of each port Tasktop utilizes and the location where you can change it if it is already being used:

| Port | Location | Purpose |
|---|---|---|
| 8080<br><br>8443 | `tasktop-hub.properties`<br>#server.port=8443<br><br>#server.redirect.port=8080<br><br>*More details here* | Default port Tasktop uses for HTTP (8080) / HTTPS (8443) |
| 8081<br><br>8444 | | User Management (Keycloak) HTTP Ports |

| | | |
|---|---|---|
| | `tasktop-hub.properties`<br>#jboss.http.port=8081<br><br>#jboss.https.port=8444<br><br>*More details here* | |
| Additional Keycloak Ports:<br><br>• 9990<br>• 9993<br>• 8009<br>• 4712<br>• 4713<br>• 25<br><br>*More details here*<br><br>(**Note**: the following ports have been modified from the Keycloak defaults: 80808081, 8443 8444) | `tasktop-hub.properties`<br>#jboss.ajp.port=8009<br><br>#jboss.management.http.port=9990<br><br>#jboss.management.https.port=9993<br><br>#jboss.txn.recovery.environment.port=4712<br><br>#jboss.txn.status.manager.port=4713<br><br>#jboss.mail.smtp.port=25 | User Management (Keycloak)<br><br>*More details here* |
| 8005 | `tasktop-hub.properties`<br>#server.shutdown.port=8005 | Tomcat Shutdown Port |

## Tasktop Hub Port

The default port Tasktop uses is 8443 for HTTPS and 8080 for HTTP, which redirects to HTTPS. If you'd like to change these ports to ease access for your users, or to accommodate a proxy, follow these instructions:

1. Open the tasktop-hub.properties file and configure the following properties:
    a. `server.port` - The http or https port
    b. `server.redirect.port` - The port that, if accessed, redirects to **`server.port`**
2. After changing the port, the address used to access Tasktop (i.e., **http://localhost:8080**) will need to be updated with the new port number in place of '8080.'

Please refer to the official documentation for additional configuration options.

To learn more about creating a `tasktop-hub.properties` file, please see the section below.

## User Management (Keycloak) Port

The default port for User Management is 8081. If you'd like to change the port that User Management (Keycloak) utilizes, follow the instructions here. If your User Management (Keycloak) utilizes a port other than 8081, you can instruct Tasktop to access User Management (Keycloak) via the correct port by following the instructions below.

1. Open the tasktop-hub.properties file and configure the following properties:
    a. `jboss.http.port` - Jboss http port
    b. `jboss.https.port` - Jboss https port

💡 **Note**: If you change the default jboss management-http port setting in the `/keycloak/standalone/configuration/standalone.xml` to something other than 9990, you must also update the port referenced in `/keycloak/bin/jboss-cli.xml`.

To learn more about creating a `tasktop-hub.properties` file, please see the section below.

# Getting Started

Once installation is complete, you can begin using Tasktop Integration Hub by opening **https://localhost:8443/** in any of our supported browsers.

Before logging on to Tasktop, you must log into the **User Administration Console** in order to create your admin user(s). The Tasktop User Administration Console can be accessed via the **User Administration Console link** at the bottom of the Tasktop Integration Hub login page. Please review the User Management section for detailed instructions on how to create a user, login, and manage your user accounts.

Once logged in, you will be prompted to set a Master Password, which will be used to encrypt your repository credentials.

You will also need to apply your license before configuring your integrations. You can learn how to apply your license here.

# Externalized Configuration

Tasktop enables you to externalize configurations from Tomcat, Jboss/Keycloak, and certain application properties in a single place. This allows you to use property files to override default values such as:

- **Jboss**: ports (e.g., http, https, management port), Keycloak database paths, Keycloak trust stores, java memory variables, and custom system properties
- **Tomcat**: ports (e.g., http https), keystores (e.g., files, passwords, types), java memory variables, and custom system properties
- **Application Properties**: Derby, Tasktop Hub, Liquibase, log4j, and keycloak host

To override default values through a properties file, you must provide the `tasktop-hub.properties` file in a directory that Hub can scan and read.

This can be done as follows:

1. Rename the file `tasktop-hub.properties.default` to `tasktop-hub.properties`.
    a. For **Windows**, this file can be found in the **App Data Directory**.
    b. For **Linux**, this file can be found in the root level of the `.tar.gz` package.
        a. **Note**: For Linux users, we recommend creating an environment variable named `TASKTOP_HOME` with its value pointing to an exclusive directory where the `tasktop-hub.properties` file will be placed.
2. Provide values to properties that need to be overridden.
    a. For example, if you'd like to change the Tomcat https port to port 9443, uncomment the property from `#server.port=8443` to `server.port=9443`

**Good to Know**:

- Only properties/lines uncommented within the `<AppDataDirectory>/tasktop-hub.properties` file will be applied, otherwise Tasktop Hub will assume default values for commented properties.
- Only properties at `<AppDataDirectory>/tasktop-hub.properties` file will be used; the file `<AppDataDirectory>/tasktop-hub.properties.default` is just a template and will not work in Tasktop Hub.

# Upgrading

## Upgrading on Windows

The `tasktop-hub.properties` file will not be replaced or deleted during the installation/upgrade process. For this reason, newer versions of Hub can retain settings automatically after upgrading.

## Upgrading on Linux

Because the properties file is placed in the `$TASKTOP_HOME` directory, newer versions of Hub will automatically apply all configurations.

If the properties file is not placed in the $TASKTOP_HOME directory, it is necessary to copy the properties file from the old installation directory to the new installation directory.

## Upgrading from a Version Earlier than 20.4

If you have made manual changes to Tomcat and/or Jboss files, you have two options upon upgrading to 20.4:

### Option One

You can apply all configurations that have been applied manually to `server.xml, standalone.xml`, `standalone.conf, standalone.conf.bat`, `setenv.sh`, and `Manage Tasktop -> Java -> Java Options` to the `tasktop-hub.properties` file .

During an upgrade, it is not necessary to override the `server.xml` file from the old version to the new installation directory. This can be done by simply providing the `tasktop-hub.properties` file in a directory that Tasktop Hub is able to read and ensuring that there is an uncommented line as shown below:

```
...
server.port=9443
...
```

💡Other properties can be configured the same way as shown in the example above.

### Option Two

You can copy all configuration files from Tomcat and/or Jboss that were previously modified and override them in the new version directories.

# Properties

The `tasktop-hub.properties` file contains three main blocks:

- Jboss/Keycloak Properties
- Tomcat Properties
- Tasktop Hub Properties

## Jboss/Keycloak

The properties listed in the table below are used only if Tasktop Hub is using Keycloak as an Authentication Provider. When provided, the properties file will be passed as an argument of `standalone.sh/standalone.bat` (e.g., `standalone.sh|bat --properties=<path>/tasktop-hub.properties`), which means that the file will override Jboss variables.

| Property | Purpose | Notes |
|---|---|---|
| | | |

| | | |
|---|---|---|
| jboss.ajp.port | Use this property to provide a value for the tag <socket-binding name="ajp" /> within the standalone.xml descriptor. | |
| jboss.http.port | Use this property to provide a value for the tag <socket-binding name="http" /> within the standalone.xml descriptor. | |
| jboss.https.port | Use this property to provide a value for the tag <socket-binding name="https" /> within the standalone.xml descriptor. | |
| jboss.management.http.port | Use this property to provide a value for the tag <socket-binding name="management-http"/> within the standalone.xml descriptor. | If this property is provided, the script <installation>/keycloak/bin/jboss-cli-tasktop.sh\|bat will call the script <installation>/keycloak/bin/jboss-cli.sh\|bat passing these arguments: --controller=localhost:**<jboss.management.http.port>** --properties=**<path>**/tasktop-hub.properties. |
| jboss.management.https.port | Use this property to provide a value for the tag <socket-binding name="management-https"/> within the standalone.xml descriptor. | |
| jboss.txn.recovery.environment.port | Use this property to provide a value for the tag <socket-binding name="txn-recovery-environment"/> within the standalone.xml descriptor. | |
| jboss.txn. | Use this property to provide a value for the tag <socket-binding name="txn-status- | |

| | | |
|---|---|---|
| `status.manager.port` | `manager"/>` within the standalone.xml descriptor. | |
| `jboss.mail.smtp.port` | Use this property to provide a value for the tag `<remote-destination host="localhost" />` within the `standalone.xml` descriptor. | |
| `jboss.server.data.dir` | Use this property if you want to place the keycloak database in a custom directory. | This is the same directory where keycloak database lives.<br><br>For both Windows and Linux, the directory separator needs to be '/'. |
| `jboss.java.memory` | Use this property to change memory settings. | We recommend setting the maximum Java heap size value to 50-75% of your system's memory. |
| `jboss.custom.system.properties` | Use this property to load custom system properties. For example: `-Djboss.*=value, -Dkey=value, -XX:key=value, -javaagent:value, -agentlib:value` | |

## Tomcat

The properties listed in the table below are used to override some properties from Tomcat.

| Property | Purpose | Notes |
|---|---|---|
| `server.port` | Use this property to provide a value for the attribute port in the tag `<Connector/>` within the `server.xml` descriptor. | After changing the port, if Keycloak is being used, you will need to go into the User Administration Console and adjust the client to the new port. |
| `server.redirect.port` | Use this property to provide a value for the attribute redirectPort in the tag `<Connector/>` within the `server.xml` descriptor. | |
| `server.shutdown.port` | Use this property to provide a value for the attribute port in the | |

| | | |
|---|---|---|
| | tag <Server/> within the server.xml descriptor. | |
| server.tomcat.connection-timeout | Use this property to provide a value for the attribute connectionTimeout in the tag <Connector/> within the server.xml descriptor. | |
| server.ssl.key-store=/path/to/keystore-file | Use this property to provide a value for the attribute keystoreFile in the tag <Connector/> within the server.xml descriptor. | This property is shared with Jboss /Keycloak. The standalone.xml file is reading this property:<br><br><spi **name="truststore"**><br><br>...<br><br><property **name="file" value="${server.ssl.key-store:${jboss.home.dir}/../insecureKeystore}"**/><br><br>... |
| server.ssl.key-store-password=changeit | Use this property to provide a value for the attribute keystorePass in the tag <Connector/> within the server.xml descriptor. | This property is shared with Jboss /Keycloak. The standalone.xml file is reading this property:<br><br><spi **name="truststore"**><br><br>...<br><br><property **name="password" value="${server.ssl.key-store-password:changeit}"**/><br><br>... |
| server.ssl.key-store-type=JKS | Use this property to provide a value for the attribute keystoreType in the tag <Connector/> within the server.xml descriptor. | |
| server.ssl.key-alias | Use this property to provide a value for the attribute keyAlias in the tag <Connector/> within the server.xml descriptor. | Enable this property only if your custom Keystore has an alias and it is different than Tomcat. |
| | | |

| | | |
|---|---|---|
| `tomcat.java.memory=-Xms256M -Xmx2118M` | Use this property to change memory settings. | We recommend setting the maximum Java heap size value to 50-75% of your system's memory.<br><br>**For Windows**: Initial memory pool size (-Xms) and maximum memory pool size (-Xmx) needs to be in MB. That means that the value needs to be suffixed with 'M'.<br><br>Values suffixed with 'G' will cause an error at the start of Hub.<br><br>**For Linux**: Values can be specified in MB or GB. Both suffixes 'M' and 'G' work. |
| `tomcat.java.endorsed.dirs=path/to/endorsed` | Use this property to provide a custom path for `java.endorsed.dirs` directory. | |
| `tomcat.java.errorFile=/path/to/hs_err_pid%%p.log` | Use this property to provide a custom path for `-XX:ErrorFile`. | |
| `tomcat.java.io.tmpdir=path/to/temp` | Use this property to provide a custom path for `java.io.tmpdir` directory. | |
| `tomcat.java.util.logging.config.file=path/to/logging.properties` | Use this property to provide a custom path for Tomcat's `logging.properties` file. | |
| `tomcat.jdk.tls.rejectClientInitiatedRenegotiation=true` | Use this property to provide `jdk.tls.rejectClientInitiatedRenegotiation` value. | |
| `tomcat.custom.system.properties` | Use this property to load custom system properties such as: | |

| | | |
|---|---|---|
| | `-XX:key=value, -javaagent:value, -agentlib:value` | |

## Tasktop Hub

The properties listed in the table below are used to override some Tasktop Hub values.

| Property | Purpose | Notes |
|---|---|---|
| `derby.storage.pageCacheSize` | Use this property to change the data page cache in the database. | Reference: https://db.apache.org/derby/docs/10.14/ref/rrefproper81359.html |
| `derby.system.home=/path/to/db` | Use this property to provide a custom path to the Derby database directory. | Providing the Derby database directory is useful for Linux environments when upgrading, as you do not need to copy files from the old installation directory to the new installation directory. |
| `hub.database.configuration.directory=/path/to/db` | Use this property to provide a custom path to the Derby database. | |
| `liquibase.ignoreRecycleBinWarning=true` | Use this property to whether or not suppress liquibase warnings. | |
| `log4j.configuration=file:/path/to/log4j2.xml` | Use this property to provide a custom path to the `log4j2.xml` file. | |
| `log4j.configuration.verbose=file:/path/to/log4j2-troubleshooting.xml` | Use this property to provide a custom path to the `log4j2-troubleshooting.xml` file. | |
| `hub.security.cors.exclusionPaths` | Use this property to provide a list of paths that will be excluded from the CORS verification. | Prior to version 21.1, this property was configured in `/tasktop/container/webapps/root/WEB-INF/web.xml` |

| | | |
|---|---|---|
| | For example: `/first-path, /second-path` | |

### Good to Know

#### Windows

- It is not possible to use environment variables to compound values. Properties related to paths must be configured using an absolute path.
- Properties must be modified in the `tasktop-hub.properties` file as this file has more priority than properties modified in `Manage Tasktop > Java > Java Options | Initial memory pool | Maximum memory pool`.

#### Linux

- It is possible to use environment variables to compound a specific value. As an example, it is possible to use $CATALINA_BASE to compound a path.

```
hub.database.configuration.directory=$CATALINA_BASE/../../directory
log4j.configuration.verbose=file:$CATALINA_BASE/../../log4j2-troubleshooting.xml
```

## Default File Locations

# Windows

When Tasktop Integration Hub is installed on Windows using the MSI installer, the program files (i.e., the executable files and binaries) are located in `C:\Program Files\Tasktop`; configuration files and logs are located in `C:\ProgramData\Tasktop`.

💡 **Tip**: ProgramData may be a hidden folder, so you will need to change your Windows Explorer settings to show hidden files and folders to find it.

⚠️ **Note**: If you change the location of the `ProgramData` directory to an alternate location, do **not** include spaces in the name of the new directory. If the directory has spaces in its name, Tasktop's UI will not be accessible.

# Linux

When Tasktop Integration Hub is installed on Linux, the program files (i.e., the executable files and binaries), configuration files, and logs are all located in the installation directory where you extracted the distribution archive.

⚠️ **Note**: You must choose a location with **no spaces** in its path, or Tasktop's UI will not be accessible.

# Repository Preparations

## Preparing Your Repositories

In Tasktop, the term **repository** refers to the external tools Tasktop connects to (e.g., Atlassian Jira, ServiceNow, BMC Remedy, etc).

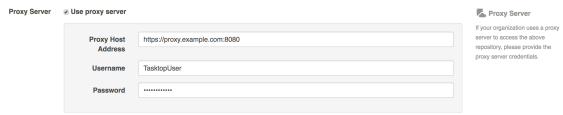
Before connecting Tasktop Integration Hub to your external repositories, you will need to perform some simple preparation on each repository you will be integrating. This preparation includes creating a user account for Tasktop Integration Hub with the appropriate permissions. Please refer to our Connect or Docs for detailed instructions for each repository.

## Firewalls and Proxies

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 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 **user proxy server** box and fill in your proxy details in the **Proxy Server** section on the New Repository Screen:



# Advanced Configuration

## Container Configuration

Tasktop is distributed with the **Apache Tomcat Servlet Container**.

For information on configuring the container, refer to Apache Tomcat documentation.

On **Windows**, configuration and log files are installed under `C:\ProgramData\Tasktop` while program files are located under `C:\Program Files\Tasktop`.

For information on configuring the service, refer to Apache Tomcat Service How to.

Further configuration, including JVM options and memory allocation, can be performed for the Windows service by launching **Tasktop Properties** located at `C:\Program Files\Tasktop\container\bin\tasktopw.exe`.

## Increasing Available Memory

Beginning in Hub version 20.4, configurations are externalized from **Tomcat**, **Jboss**/**Keycloak**, and certain application properties in a single place. This allows you to use property files to override memory variables and custom system properties.

For more information on changing memory settings, please refer to the properties table.

## Logging

Logging is configured with `log4j2`. See the included `log4j2.xml` to configure log levels, location, and rolling policy.

The included `log4j2-troubleshooting.xml` configures `log4j2` for the troubleshooting log level when set via the Settings screen.

# Upgrading

Beginning in the Tasktop Hub 20.4 release, a properties file is used for configurations such as Ports & SSL. To learn more about this update, click here.

Please be aware of additional upgrade steps needed for the scenarios outlined below.

**When upgrading from earlier versions to 21.1.x:**

- As 21.1.x is a checkpoint release, be aware that you **must** upgrade to version 21.1.x **before** upgrading to a later version.
- If you do not follow instructions here, you may encounter issues with artifact association management that prevent you from viewing and deleting artifact pairs.
- **Note**: Be aware that upgrading to 21.1.x from an earlier version may take longer than usual.

**When upgrading from a version earlier than 19.3.0.20190603:**

- If you do not follow instructions here, you may experience errors that prevent pages from loading or be unable to log out of Tasktop.
- If you are upgrading from a version that is ***also earlier than 19.2.1***, *please follow the additional instructions below:*
    - While we always recommend backing up the operational database, it is imperative that a backup is made prior to upgrading to 19.2.1 or later. Upon upgrade from a version earlier than 19.2.1 to 19.2.1 or later, a one-time change to the operational database will occur that may take an hour or longer to complete. During the upgrade process, the UI will not be available. To monitor the upgrade process, please inspect the log files. You can find more details in our FAQ here.

**When upgrading from earlier versions (e.g., 18.3 and earlier) of Hub to newer versions:**

- You may need to perform a two-step upgrade to prevent an upgrade failure.

## Backup

A working backup strategy is a critical element of disaster recovery, since only backups can mitigate complete hardware failure and user error. A strategy that ensures correct and current backups is essential. Backups of the Tasktop database include both configuration and operational data.

Backup frequency should mirror your practices for all software tools your organization utilizes. Backup frequency should be daily, ideally with incremental backups performed more frequently.

# General Application Configuration

The recommended practice is to back up the entire installation/program data directory to cover all customizations (excluding logs)

- Back up Tomcat customizations (in Linux install directory or Windows Program Data)
  - `container/conf/server.xml`
  - Any keystores for certificates
  - For **Linux**: `bin/setenv.sh`
  - For **Windows**: any changes to the Java section of the Manage Tasktop application (e.g., memory, command line parameters, etc)
- Back up keycloak data and customizations
  - `keycloak/standalone/data`
  - `keycloak/standalone/configuration/standalone.xml`

# Operational Data

## Default Derby Database

⚠️ Tasktop automatically stores operational data to a built-in database. However, for production environments, we strongly recommend that operational data is stored to an external database for improved maintainability. This enables you to perform frequent backups without stopping Tasktop Integration Hub and ensures that your Tasktop practices are consistent with your existing disaster and recovery process. For details on how to store your operational data to an external database rather than Tasktop's built-in database, please refer to General (Settings).

If utilizing Tasktop's built-in Derby Database, ensure you've backed up the following:

- File backup of db directory (in Linux install directory or Windows Program Data)

## External Database

In order to back up Tasktop Integration Hub, follow the instructions below:

1. Ensure that you have migrated your operational data to an external database. For details on how to set up your external database, please see General (Settings).
2. Back up the following folders
   a. on **Linux**:
      i. `/tasktop/db`
      ii. `/tasktop/drivers`
      iii. `/tasktop/libraries`
   b. on **Windows**:
      i. The Tasktop data folder, typically `C:\ProgramData\Tasktop`
3. Back up the external database using that database's backup tools.
4. Back up the Tomcat and Catalina configuration

💡 **Note**: This is only applicable if changes are made to the Tomcat and Catalina configuration.

# Restore from Backup

If Tasktop fails to restart after an upgrade or if there are unresolvable errors preventing your integrations from running, Tasktop may need to be returned to the previous version. Please ensure to stop Tasktop before restoring to a previous version.

⚠️ **Note**: If restoring from backup, you should be cautious as the state of the integration is maintained in the database and restoring to an older version could result in duplicated items and data (e.g., comments and attachments). It is recommended to only restore when directed by Tasktop support or after a failed upgrade where no items were processed.

💡 **Tip**: If integrations were resumed individually during an upgrade, you can prevent duplicating items and data when restoring to an earlier version by utilizing the upgrade from backup file feature described below.

## General Application Configuration

You should restore any changes identified in the backup.

## Operational Data

### Default Derby Database

In order to restore Tasktop Integration Hub, follow the instructions below:

1. Copy the database directory from backup to the Tasktop data folder.
2. Restore the backed up Tomcat and Catalina configuration files from part 4 of the backup instructions.

### External Database

In order to restore Tasktop Integration Hub, follow the instructions below:

1. Restore the external database backup using the tools from that database.
2. Restore the backed up Tomcat and Catalina configuration files from part 4 of the backup instructions.

## Operating Systems

### Windows

1. Shut down Tasktop.

2. Uninstall Tasktop, then run the previous installer.
3. Restore from backup as described in section above.
4. Restart Tasktop.
    a. If you'd like, you can restart Tasktop in 'safe mode' which ensures all integrations are paused.

### Linux

1. Shut down Tasktop.
2. Remove the new Tasktop installation folder and restore the old Tasktop installation folder from step 3 of the upgrade steps.
3. If you are using an external database for Tasktop's configuration, restore the external database as described above.
4. Restart Tasktop.
    a. If you'd like, you can restart Tasktop in 'safe mode' which ensures all integrations are paused.

# Upgrading

## Before you Upgrade

⚠️ Before upgrading Tasktop, be sure to do the following:

1. Shut down Tasktop and afterwards follow the backup instructions outlined above. The first time that Tasktop restarts after an upgrade, the internal database will be migrated to the new version and it will no longer be possible to return to the prior version without the backup.
2. Additionally, ensure that backups are made of the Tomcat, Catalina, and Keycloak configuration files that have been customized. The upgrade process will overwrite these configuration files and customizations will need to be re-applied.
3. When Tasktop is upgraded, a service-downtime for the Tasktop service is required in order to upgrade the database. Note that a second instance cannot be running while the first instance is attempting to upgrade the database.
    a. To understand implications of Tasktop downtime, please see here.
4. Please review the release notes for all Tasktop versions that have been released after the version you are upgrading from. Ensure that any upgrade steps outlined in the release notes are followed.

### Windows

1. Ensure a copy of the old installer is available in case a roll-back is required.
2. Click the **Stop Tasktop** button on your desktop, and make sure services are stopped:

3. Backup as described in section above.
4. Run the installer of the new version of Tasktop.
5. Re-apply Tomcat/Keycloak configurations.

a. Upgrading from versions *earlier* than 20.4:
    i. Apply all customizations done in (`<install-location>/container/conf/server.xml`) to the `tasktop-hub.properties` file. More details about translating configurations from `server.xml` to the new properties file can be found here.
    ii. Apply all customizations that have been done in (`<install-location>/keycloak/standalone/configuration/standalone.xml`) to the `tasktop-hub.properties` file. More details about translating configurations from `standalone.xml` to the properties file can be found here.
b. Upgrading from versions *earlier* than 21.1:
    i. If any configuration was applied to `exclusion-paths` property in the `web.xml`, it needs to be migrated to the tasktop-hub.properties file. See the following example:
        1. Copy `/auth/realms/Tasktop/broker/saml` and `/auth/realms/Tasktop/login-actions` from the `web.xml` file.
            a.
```
<filter>
    <filter-name>CORSFilter</filter-name>
    <filter-class>com.tasktop.servlet.cors.CorsHeaderScrutinyServletFilter<
/filter-class>
    <init-param>
        <description>A comma or whitespace separated list of paths to
exclude from the CORSFilter</description>
        <param-name>exclusion-paths</param-name>
        <param-value>
            /auth/realms/Tasktop/broker/saml
            /auth/realms/Tasktop/login-actions
        </param-value>
    </init-param>
</filter>
```
        2. Place them in the `tasktop-hub.properties` file.
            a.
```
# A list of paths that will be excluded from the CORS verification.
# This list is separated by comma. Example: /first-path,/second-path
hub.security.cors.exclusionPaths=/auth/realms/Tasktop/broker/saml,/auth
/realms/Tasktop/login-actions
```
c. Upgrading from version 20.4 and later:
    i. No action needs to be taken. Tomcat/Keycloak configurations will be applied automatically.

6. If you have connected to the Microsoft TFS repository in the past:
    1. Remove all files and folders, except for the com.tasktop files, under `<install-location>\Tasktop\libraries\microsoft-tfs` and `<program-data>\Tasktop\libraries\microsoft-tfs`. Note that the parent folders (marked in red here) for each location could differ if they were customized during original installation.
    2. Once Tasktop is started up again, navigate to the TFS repository connection screen. There, you will see instructions on how to provide the updated SDK and CLC files to Tasktop by adding them to the connector-requirements directory on the machine that hosts Tasktop.
    3. Restart Tasktop after uploading the files.
7. Start Tasktop.
8. Navigate to the Activity screen.
    a. Review the Background Jobs tab to review status on Integration Data Migration jobs.
        i. Resolve any associated issues shown here. These issues will need to be resolved and their associated data migration jobs completed before the affected integrations can run (unrelated integrations should continue to process).

ii. Once the associated issue is resolved, failed Integration Data Migration processes can be prioritized using the 'prioritize' button on the Background Jobs tab. Ensure that these jobs complete successfully.

b. Review the Issues tab to resolve any configuration issues.
   i. If there are configuration migration issues, those will be shown in the Issues tab. They will block affected integrations from running (but unrelated integrations should continue to process). Once the source of the issue is resolved, configuration migration issues can be retried using the 'retry' button on the Issues tab.
   ii. If using TFS, you may see issues related to unsatisfied connector requirements since you may need to upload new versions of the TFS SDK and CLC zip files.

c. Review the Errors tab to resolve any errors related to specific integration activities.

d. Once all issues and errors are resolved, the internal upgrade will complete and information will begin processing for those affected integrations.

9. If you are upgrading from a version earlier than 19.4.1, please see details regarding the Troubleshooting User here.

## Linux

1. Shut down Tasktop and Keycloak.
2. Back up as described in section above.
3. Move the old Tasktop installation folder to an archive folder.
4. Unzip the new Tasktop distribution archive.
5. Restore drivers, copy the `/tasktop/drivers` directory from the old installation into the new installation folder `<install-location>/tasktop`.
6. Restore DB.

   a. If you are using Tasktop's internal configuration database, copy the `tasktop/db` folder from the old installation into the new installation folder `<install-location>/tasktop`.
   b. If you are using an external database for Tasktop's configuration, copy the `tasktop-db.json` file, and the `/tasktop/db` from the old installation into the new installation folder `<install-location>/tasktop`.

7. Re-apply Tomcat/Keycloak configurations.

   a. Upgrading from versions *earlier* than 20.4:
      i. Apply all customizations done in (`<install-location>/container/conf/server.xml`) to the `tasktop-hub.properties` file. More details about translating configurations from `server.xml` to the properties file can be found here.
      ii. Apply all customizations done in (`<install-location>/keycloak/standalone/configuration/standalone.xml`) to the `tasktop-hub.properties` file. More details about translating configurations from `standalone.xml` to the properties file can be found here.
   b. Upgrading from versions *earlier* than 21.1:
      i. If any configuration was applied to `exclusion-paths` property in `web.xml`, it will need to be migrated to the `tasktop-hub.properties` file. See the following example:
         1. Copy `/auth/realms/Tasktop/broker/saml` and `/auth/realms/Tasktop/login-actions` from the `web.xml` file:

a.
```
<filter>
    <filter-name>CORSFilter</filter-name>
    <filter-class>com.tasktop.servlet.cors.CorsHeaderScrutinyServletFilter<
/filter-class>
    <init-param>
        <description>A comma or whitespace separated list of paths to
exclude from the CORSFilter</description>
        <param-name>exclusion-paths</param-name>
        <param-value>
            /auth/realms/Tasktop/broker/saml
            /auth/realms/Tasktop/login-actions
        </param-value>
    </init-param>
</filter>
```

2. Place them in the tasktop-hub.properties file:
   a.
```
# A list of paths that will be excluded from the CORS verification.
# This list is separated by comma. Example: /first-path,/second-path
hub.security.cors.exclusionPaths=/auth/realms/Tasktop/broker/saml,/auth
/realms/Tasktop/login-actions
```

c. Upgrading from version 20.4 and later:
   i. If any customization has been applied to the `tasktop-hub.properties` file, copy it into the new installation folder `<install-location>/tasktop`.

8. Restore Keycloak (user management) configuration. Note that keycloak's database and Tasktop's database are separate.
   a. If you are using Keycloak's internal configuration database, restore the database (`<install-location>/keycloak/standalone/data/keycloak.h2.db`) after installation.
   b. If you are using an external database for Keycloak's configuration, reconfigure the external database as described here
      a. 💡 **Note**: You must create an account to access these.
9. If you have connected to the Microsoft TFS repository in the past:
   a. Remove all files and folders, except for the com.tasktop files, under `<install-location>\Tasktop\libraries\microsoft-tfs.`
   b. Once Tasktop is started up again, navigate to the TFS repository connection screen. There, you will see instructions on how to provide the updated SDK and CLC files to Tasktop by adding them to the connector-requirements directory on the machine that hosts Tasktop.
   c. Restart Tasktop after uploading the files.
10. Start Tasktop.
11. Navigate to the Activity screen.
    a. Review the Background Jobs tab to review status on Integration Data Migration jobs.
       i. Resolve any associated issues shown here. These issues will need to be resolved and their associated data migration jobs completed before the affected integrations can run (unrelated integrations should continue to process).
       ii. Once the associated issue is resolved, failed Integration Data Migration processes can be prioritized using the 'prioritize' button on the Background Jobs tab. Ensure that these jobs complete successfully.
    b. Review the Issues tab to resolve any configuration issues.

i. If there are configuration migration issues, those will be shown in the Issues tab. They will block affected integrations from running (but unrelated integrations should continue to process). Once the source of the issue is resolved, configuration migration issues can be retried using the 'retry' button on the Issues tab.

c. Review the Errors tab to resolve any errors related to specific integration activities.

d. Once all issues and errors are resolved, the internal upgrade will complete and information will begin processing for those affected integrations.

12. If you are upgrading from a version earlier than 19.4.1, please see details regarding the Troubleshooting User here.

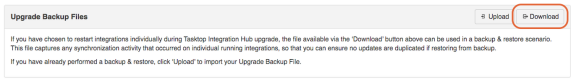
# Upgrade from Backup File

*This feature is not applicable to Tasktop Cloud and is only available when upgrading from Tasktop Integration Hub versions 20.1 and later. To utilize this feature, please see the section below.*

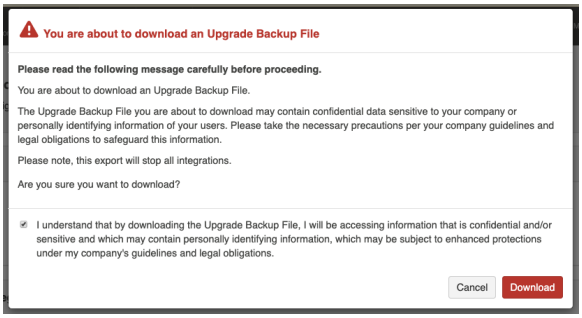
To restore Hub to a previous version in cases where integrations were resumed individually during an upgrade, you must use the upgrade backup file available on the **Advanced Configuration** screen. The downloaded data in the file corresponds to artifacts that may have been modified when migrations were still running to ensure artifact updates aren't duplicated when restoring.

⚠ **Note**: You must download the backup file from your Hub instance **before** beginning the steps to restore.

To use this feature, you will first need to download the upgrade backup file. This file can be downloaded on the **Advanced Configuration** screen.

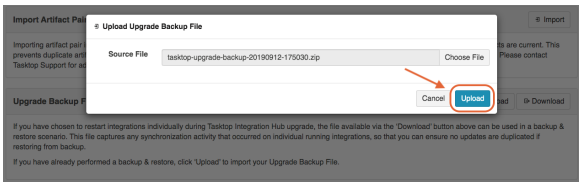



After clicking **Download**, the following message will appear:

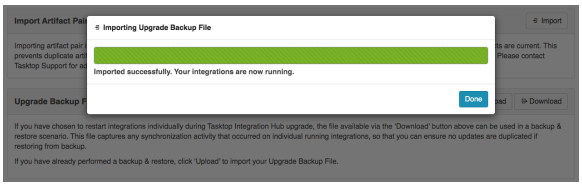



Once the file has been downloaded, you will need to restore Tasktop to the prior version. Please see the section above for more details on how to restore to a prior version.

After restoring to the earlier version, you can then select the backup file you would like to import and click **Upload**.




If the backup file is imported successfully, the following message will appear and your integrations will resume.




⚠ **Note**: If the backup file fails to upload, you will need to contact Tasktop support for further assistance.

# Business Continuity

## Overview

Tasktop Integration Hub maintains information critical to organizational business processes, and therefore should be included in a comprehensive business continuity plan that safeguards data and ensures business continuity in hardware and operational failure scenarios.

💡 For additional information, please contact Tasktop Support or your Sales Rep to access our Business Continuity & Disaster Recovery materials.

## Data Loss Prevention

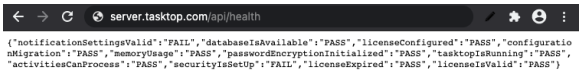
An important aspect of disaster avoidance is avoidance of data loss. Tasktop Integration Hub should be configured to use a reliable external database such as Oracle or Microsoft SQL Server. Please see the Supported Databases for storing Tasktop Operational Data section to determine supported databases.

External databases should be set up with sufficient redundancy to maximize uptime and to reduce the probability of data loss due to hardware failure. For details on how to set up your external database, please see our General (Settings) screen.

## Monitoring

You can append `/api/health` to your Tasktop URL (e.g., https://server.tasktop.com/api/health) to get information on general health of your Tasktop instance (e.g., to confirm that Tasktop is not experiencing downtime or that your license is valid).

Customers may wish to leverage this API call into a monitoring tool to allow them to determine if a failover instance need be brought up in case of issues.




Below is a definition of what each term means:

- **notificationsettingsValid**
  - *Pass:* Testing the connection to the email server succeeded.
- **databaseisAvailable**
  - *Pass:* Connecting to the operational database succeeded.
  - *Fail:* Tasktop could not connect to the Operational Database; Tasktop cannot function until this is resolved.
- **licenseConfigured**
  - *Pass*: Tasktop has been configured with a license
- **configurationMigration**

    - *Pass*: No errors from configuration migration are present, i.e., configuration migration completed successfully the last time it ran.
- **memoryUsage**
    - *Pass*: No "out of memory" errors are present.
- **passwordEncryptionInitialized**
    - *Pass*: No cryptography errors (error type CCRRTT-60001) exist (i.e., the Java runtime environment supports 256-bit AES encryption).
- **tasktopIsRunning**
    - *Pass*: Tasktop has initialized and is running, meaning the UI should be accessible. Tasktop is not currently restarting or shutting down.
    - *Fail*: Tasktop is initializing, restarting, or shutting down.
- **securityIsSetUp**
    - *Pass*: Tasktop has been configured with a master password, and the master password has been entered if necessary.
    - *Fail*: Either the master password has not yet been set up, needs to be re-entered, or Tasktop has been configured in insecure mode (no longer supported or possible to configure).
- **licenseExpired**
    - *Fail*: The license has expired.
- **licenseIsValid**
    - *Pass*: All configured integrations are allowed by the configured license.
    - *Fail*: There is no license, or there is an integration whose integration style is not licensed, or there is an integration using a connector that is not licensed.
- **activitiesCanProcess**
    - *Pass:* All valid integrations can detect changes and process activity.
    - *Fail:* Tasktop is not detecting changes or processing activity for any of the *valid Work Item or Work Item + Container integrations. Check for any error messages in Hub to resolve the issue.
        - ***Note**: Valid integrations are the ones not blocked by issues caused by user configuration errors or external factors (e.g., unavailable repositories).

# Downtime

When Tasktop service is unavailable, changes may be taking place in integrated repositories. Normal Tasktop operation ensures that data flows between these repositories in a timely manner. When the server is unavailable, however, information is no longer propagating between integrated systems.

This has the following impacts:

1. Synchronization integrations will not create or update artifacts in synchronized repositories

2. Enterprise Data Stream integrations will not record artifact changes from their integrated source repositories to their target databases, which may cause a loss of fidelity in reporting data
3. Gateway integrations cannot accept payloads from integrated gateway collections; this can result in data loss if the integrated tools cannot handle the downtime

Upon restarting Tasktop Integration Hub, integrations will resume with the following effects:

1. All Synchronization integrations will begin processing where they left off when the server became unavailable; there may be a backlog of changes to process, but no synchronizations will be lost
2. Enterprise Data Stream integrations will begin detecting artifact changes; any changes that occurred when service was unavailable will be detected, but multiple changes to the same field will have lost fidelity (only one change to that field will be reported)
3. Tasktop will begin accepting Gateway collection payloads, and if the integrated repositories are configured correctly to retry payloads, they will be processed as usual without data loss

### Backup

A working backup strategy is a critical element of disaster recovery, since only backups can mitigate complete hardware failure and user error. A backup strategy that ensures correct and current backups is essential. Backups of the Tasktop database include both configuration and operational data.

See details on Backup procedures in the Upgrading section.

## Restore

In order to restore Tasktop Integration Hub, follow the instructions outlined in the Upgrading section.

## High Availability

To learn more about Tasktop High Availability strategies, please reach out to Tasktop Support or your Sales Rep to access our Business Continuity & Disaster Recover materials.

## Load Balancing

To learn more about Tasktop's recommendation for handling REST API traffic to a repository, see our FAQ page.

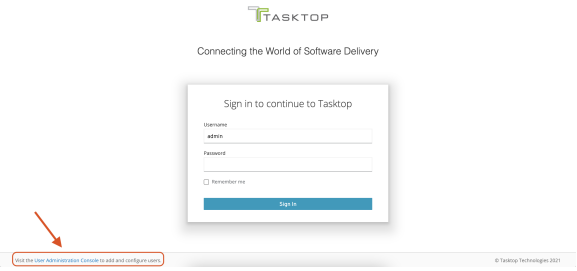
# User Management

## Getting Started

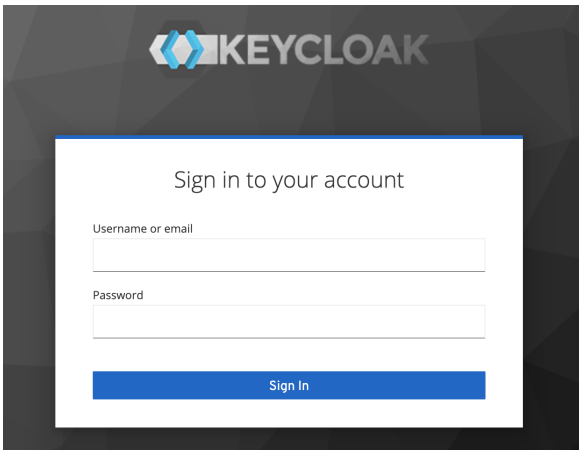
***Tasktop Cloud users will access user administration directly via the Tasktop UI and not in the external User Administration Console.***

Once installation is complete, you can begin using Tasktop Integration Hub by opening **http://localhost:8080/** or **https://localhost:8443/** in any of our supported browsers.

Before logging in to Tasktop Integration 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.



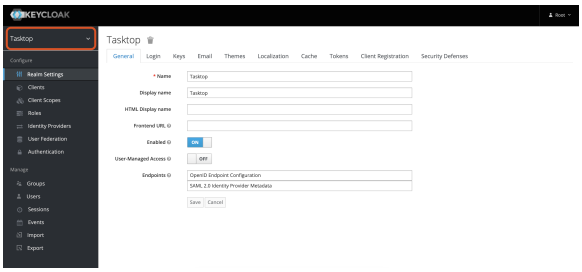

The Tasktop User Administration Console comes pre-configured with a root user and password. You can use the following credentials to log in to Keycloak:

- **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.

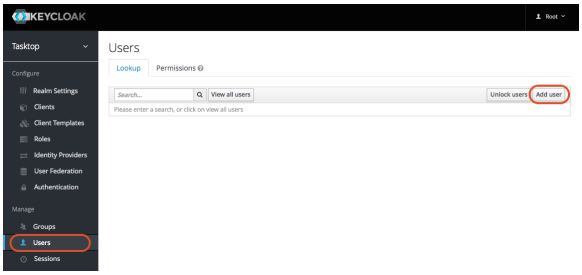
After logging in, you will be prompted to change your root password and you will need to make at least **one** new Tasktop Admin user for Tasktop. After this first user is created, you can create additional users directly from the Tasktop interface.

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

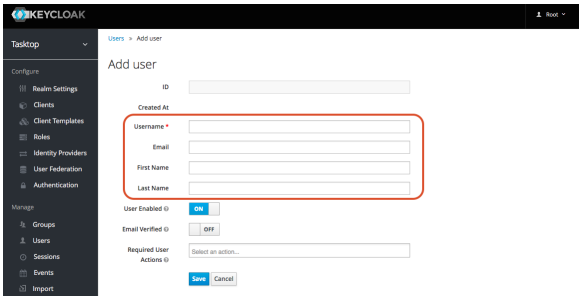



⚠️ **Note**: Do not rename the realm (**Tasktop**), as this will result in errors upon Tasktop log in. If you must rename the realm, please also edit `{tasktop workspace}/webapps/ROOT/WEB-INF/keycloak.json`, update the 'realm' parameter, and then restart Tasktop.

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



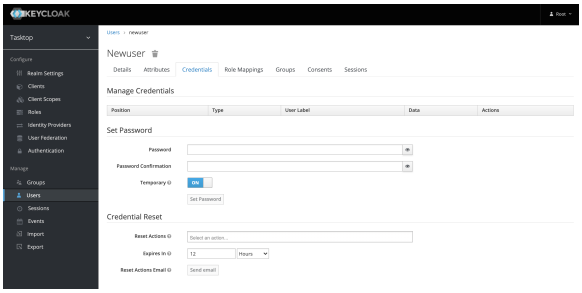

On this screen, enter the username, email, first name, and last name for your new user — the rest of the fields are **not** required. Once you've entered the required fields, click **Save**.



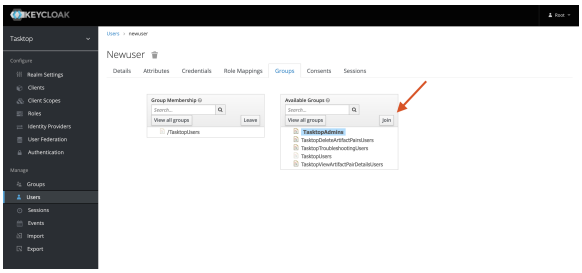

After you have saved the user, select the **Credentials** tab and provide a temporary password for the new user.

💡 **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 Integration 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.

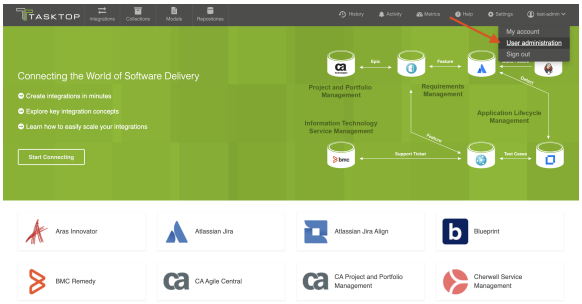
### User Permissions

| 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 */api/v1/integrations/delete-integration-data* public API | ✓ | ✗ | ✗ | ✗ | ✗ |
| Access to */api/v1/integrations/delete-all-integration-data* public API | ✓ | ✗ | ✗ | ✗ | ✗ |

# Creating Additional Users

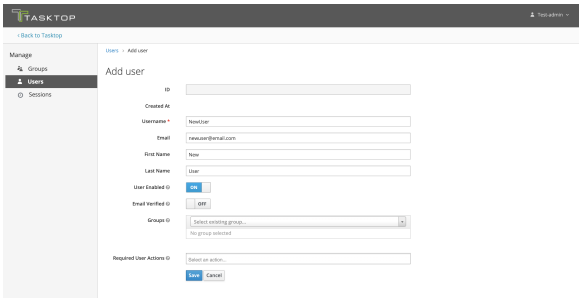
To create a user, select **User Administration**.

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




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

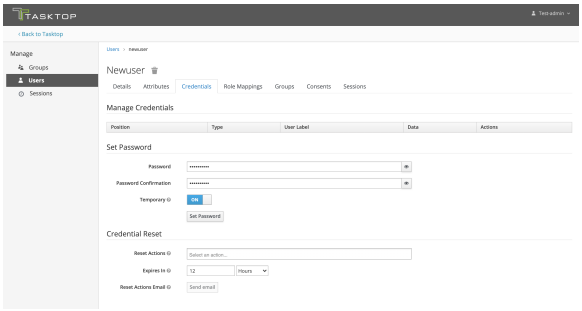

On the **Add User** screen, enter the username, email, first name, and last name for your new user — the rest of the fields are **not** required. Once you've entered the required fields, click **Save**.




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

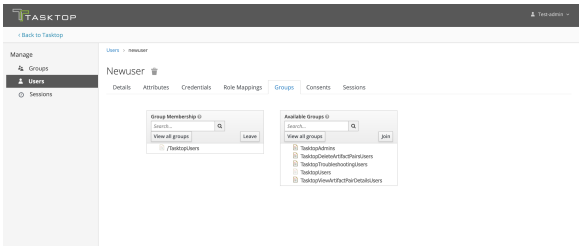
💡 **Note**: Please ensure the Temporary toggle is set to **On**. This will allow the user to set a new password upon their first log in.

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 Integration Hub.




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

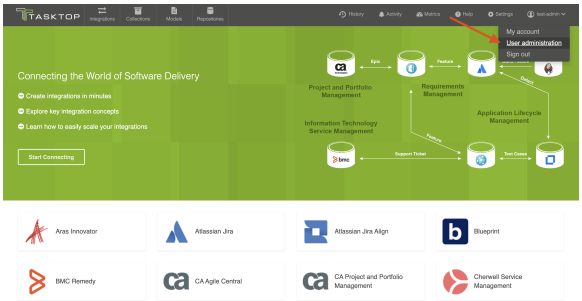
That's it! Your user has been added and can log in with their temporary password.

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

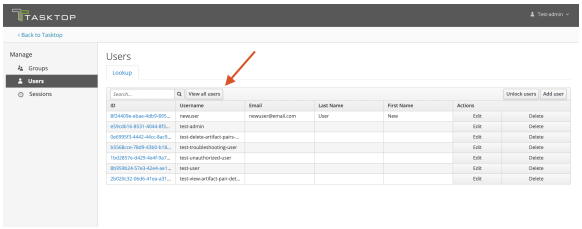
# Resetting a User's Password

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.



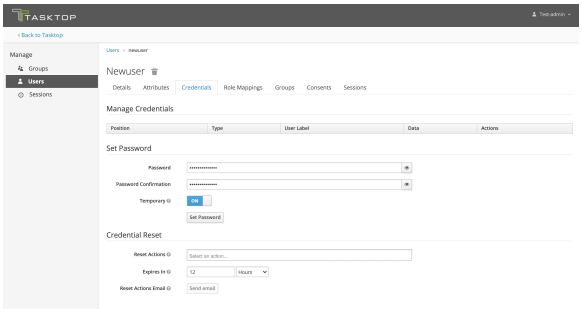

Click **View all users**. Next, click on the **ID** for the user whose password you'd like to reset.



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**.



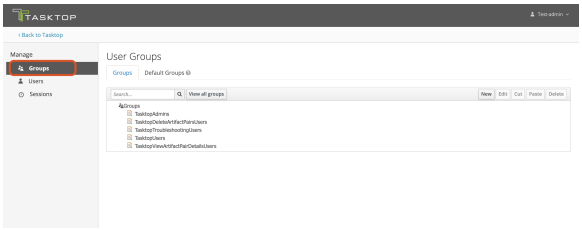
⚠️ **Note**: Tasktop will not send the user an email notification. The **admin** must notify the user of the new temporary password. The user will be prompted to set a new password upon their next log in.

# Managing Groups

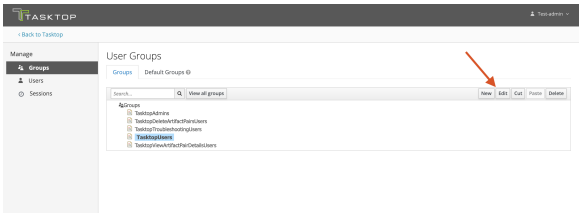
## Viewing Members of a Group

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.

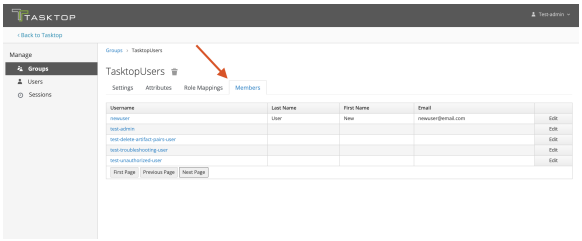


Next, select the group you'd like to review, and click **Edit**.




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

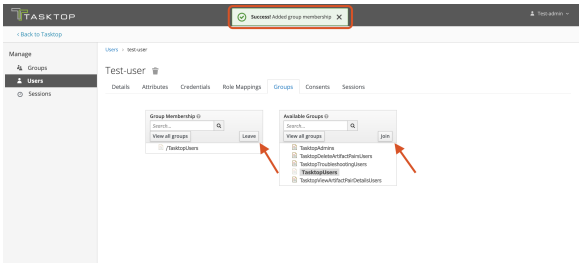
Select **Users** from the left pane of the **User Administration** screen. Click **View all Users** and select the **ID** of the user you'd like to modify.

💡 **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.

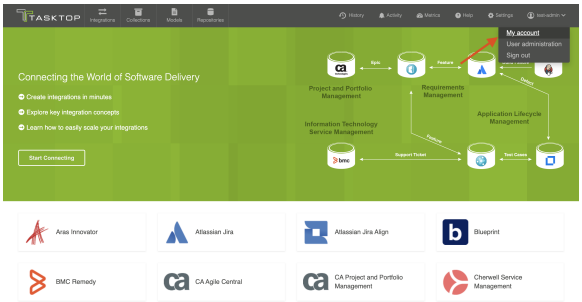
⚠️ **Note**: A user must be a member of at least **one** group in order to be able to log in to Tasktop successfully.



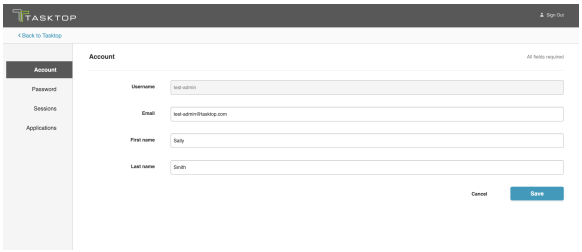

# Modifying Your Own User Information

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

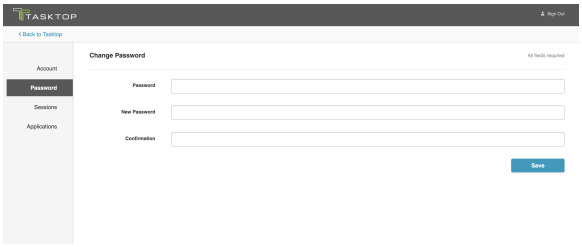
💡 **Tip**: Both **users** and **admins** can modify their own account information.




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.

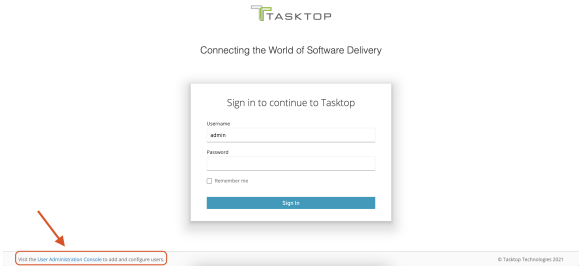



💡 **Tip**: The **Sessions** and **Applications** sections can be ignored.

# Advanced User Management

Tasktop Integration 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.

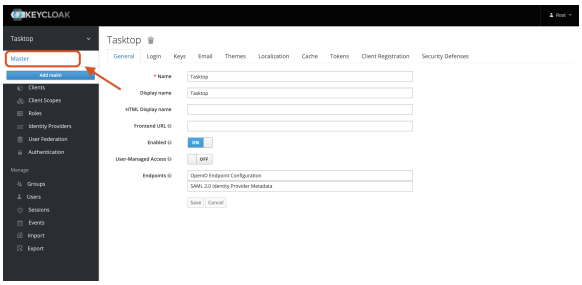
To learn more about these advanced features, click here.

⚠ **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 Integration 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.

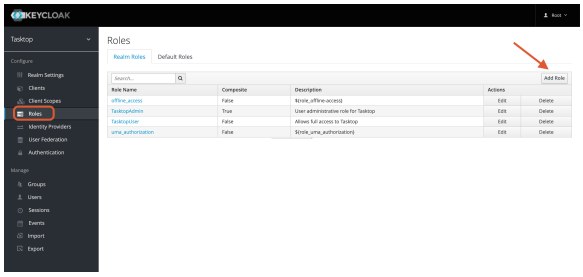
#### Linux

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

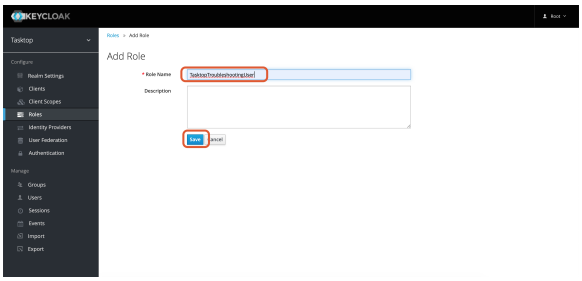
If you have not provided a valid SSL certificate, you can create a troubleshooting user via the **User Administration Console**. This console can be accessed by following the instructions in the Getting Started section.

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.

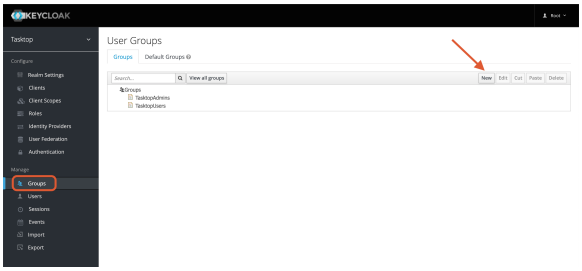



That's it! The troubleshooting user role has been created. Next, you'll need to add the troubleshooting user to a group.

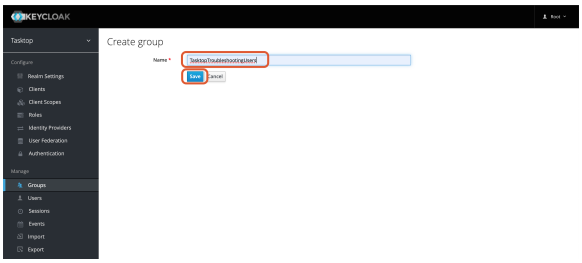
### Adding Troubleshooting Users to a Group

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**.

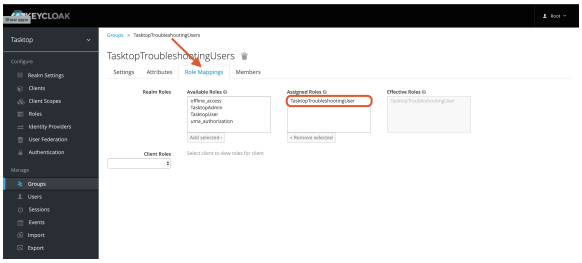



On the Create Group screen, enter **TasktopTroubleshootingUsers** in the Name field. Then, click **Save**.



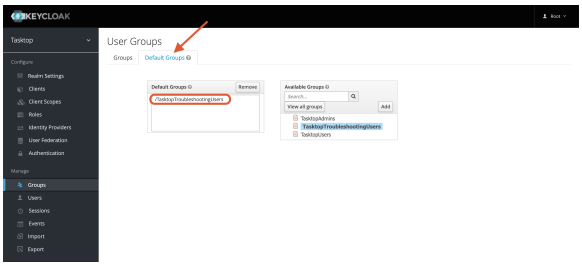

After saving the group, the new group screen will appear.

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 Integration 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.

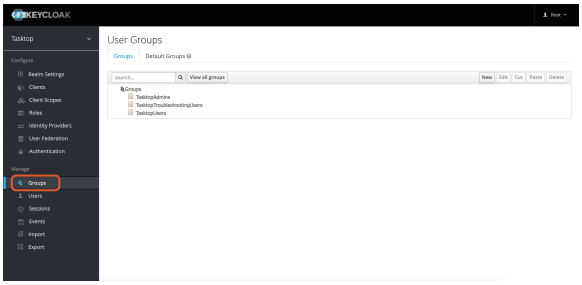
### Linux

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

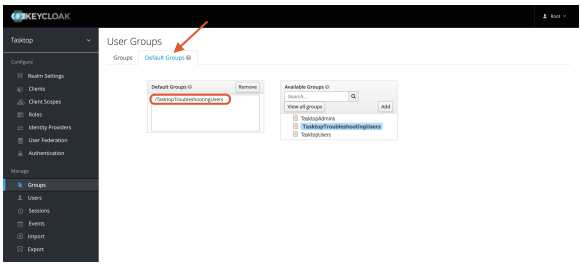
## Setting the Default Troubleshooting User Group via the Keycloak Admin Console

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 **TasktopTroublesh ootingUsers**.



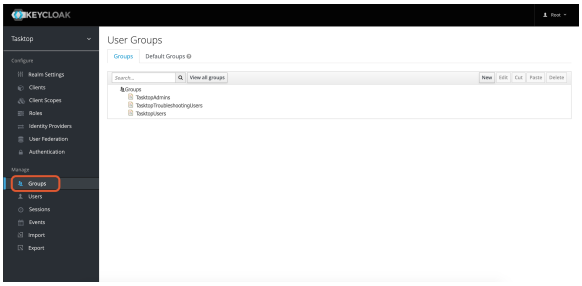

This section is only applicable to Tasktop Integration Hub version 21.2 and later.

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.

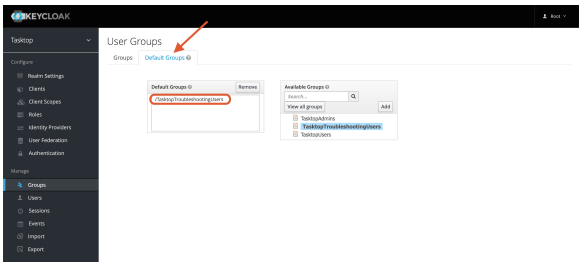
## Setting the Default Troubleshooting User Group via the Keycloak Admin Console

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 **TasktopTroublesh ootingUsers**.




### 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

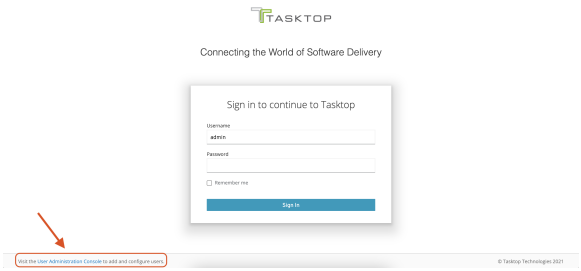
If you would like to connect to an LDAP server, you will need to import the SSL certificate into the keystore of your Tasktop product and restart it. To import the certificate to the keystore, see the following:

- 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

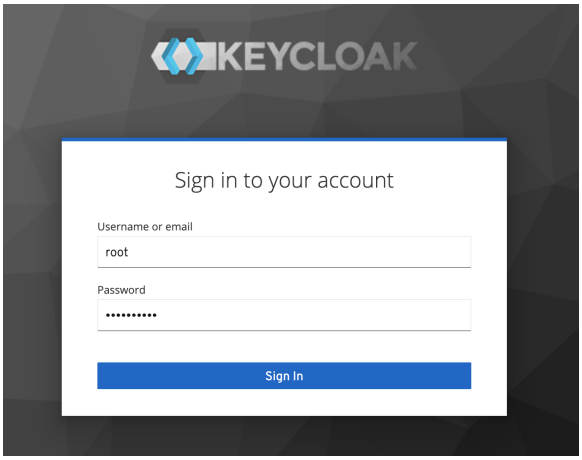
    - echo -n | openssl s_client -connect <ldap-server>:636 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > ldapserver.pem
- In command prompt enter the following:
    - <path_to_jre>/bin/keytool -import -trustcacerts -keystore <path_to_keystore> -storepass <password> -alias ldap -import -file ldapserver.pem
        - <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 changeit 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

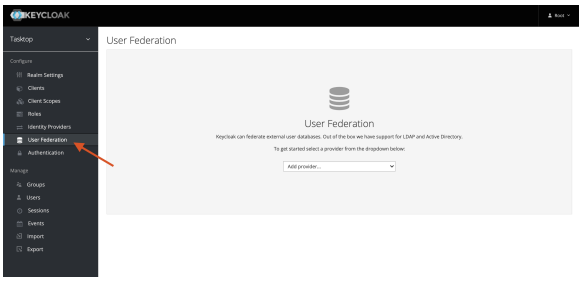
1. To access advanced user management capabilities, click the **User Administration Console link** at the bottom of the Tasktop Integration Hub login screen.



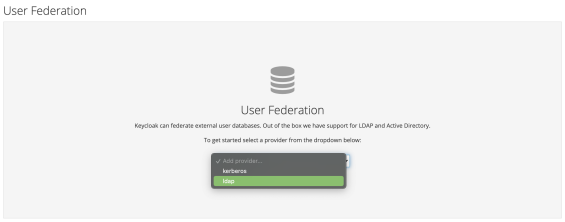

2. Log in using the default credentials listed in the Getting Started section above.



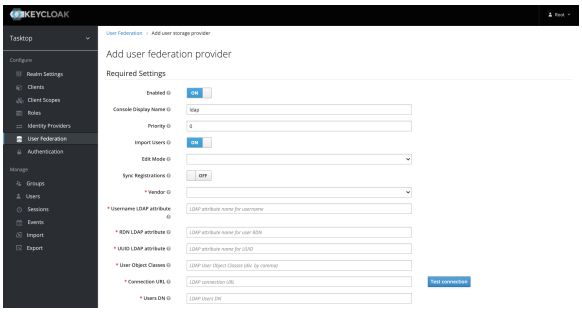

3. Select the **User Federation** link from the left side panel.




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

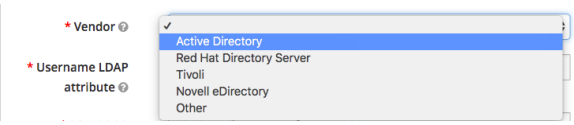
💡 **Tip**: Follow the steps above to access the **LDAP configuration** page.

See the required settings below:

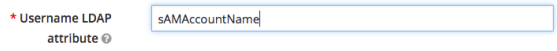
- **Console Display name**: This is the name you'd like to give your connection.
- **Priority**: If you have more than one User Federation configured, this setting specifies in which order to search each user federation service, **0** is first.
- **Edit Mode**:
  - **READ_ONLY**: This setting reads the 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**.

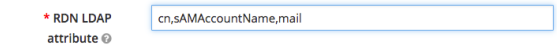
- **Sync registrations**: If a new user is created in Tasktop, this will allow that user to also be created in AD if you have **WRITABLE** selected and access to create user objects in the AD domain. The default setting is **OFF**.
- **Vendor**: Specifies which vendor software to use for this LDAP configuration. If you are using something other than Active Directory, the attributes and locations may be different. This will also pre-fill some default values.



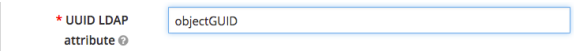

- **Username LDAP attribute**: This should be the default username attribute as specified in your domain. The default for Microsoft AD is **sAMAccountName**.



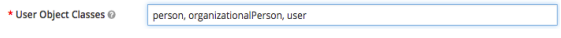

- **RDN LDAP attribute**: The Relative Distinguished Name LDAP attribute is a list of attributes which will be searched when a user attempts to authenticate to Tasktop. The attributes listed here should be unique within an OU level or unique within a domain. The following options are a good base to use:
  - **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*)




- **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**.




- **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**.



- **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).


**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.

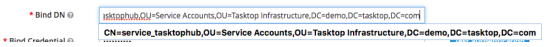
- **Users DN**: This is the Distinguished Name for the location where you can find your users. You can find the Users DN (and any other Distinguished Names) via the **ADSIEdit** tool in Windows. Once the tool is open, you will need to connect to the AD domain for your organization. Once connected, the 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.


- **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.

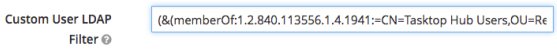



- **Bind Credential**: This is the password for the user account configured in the Bind DN.

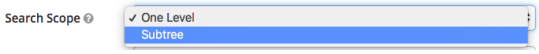
**Tip**: Once you have entered the password, click **Test authentication** to confirm that Tasktop is successful in authenticating itself against your Active Directory domain. You should see a green message at the top of your page as an indication of a successful authentication.

- **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 **U sername**. 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 in 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.

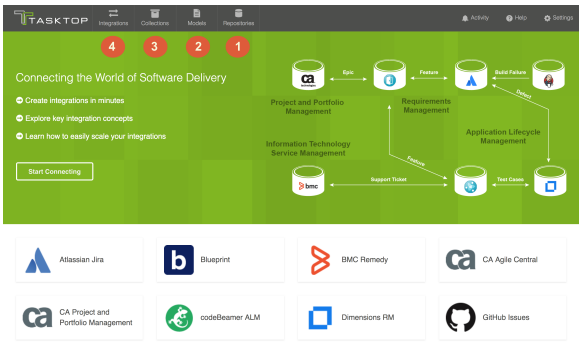
Whether you're just getting started with Hub or need a quick refresher on how to create an integration, read this quick-start guide for a summary of each step and click the links to learn more.

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**.




---

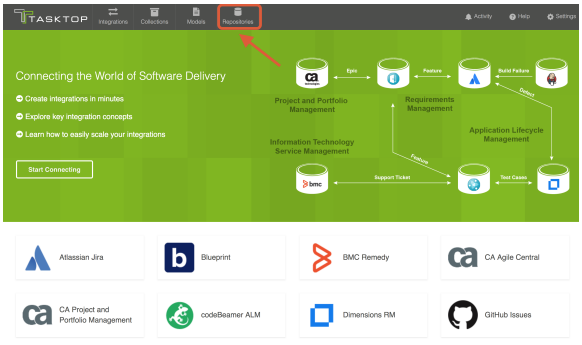
## Connect to your Repository

The first step to take when configuring an integration is to connect to your **repository**.

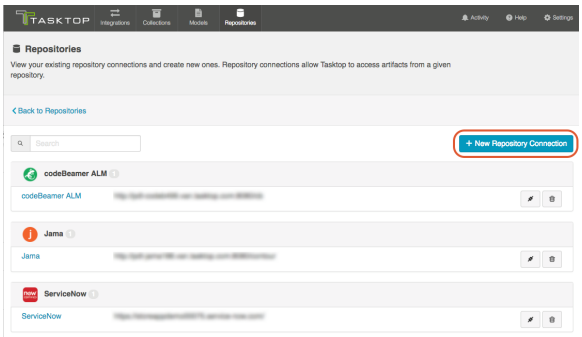
Your repositories refer to the external tools that Tasktop will flow information between.

You can create two types of repository connections: **Standard** and **Database**.

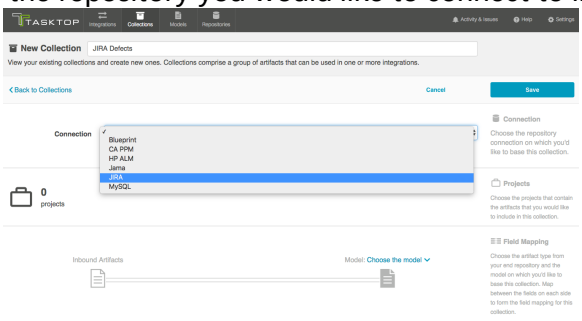
1. To create a new repository connection, select **Repositories** at the top of the screen.

a. 

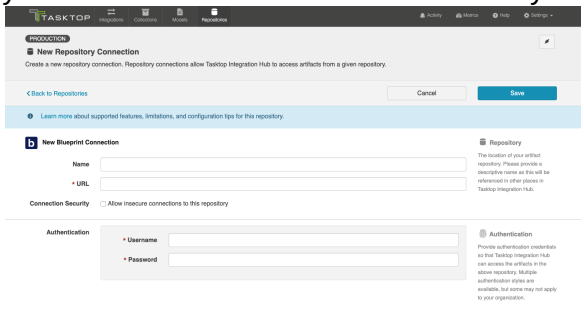

2. Click **+ New Repository Connection**.

a. 


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

a. 


4. Enter your authentication credentials and save your connection.

a. 


For detailed instructions on connecting to Standard and Database repositories, click **here**.

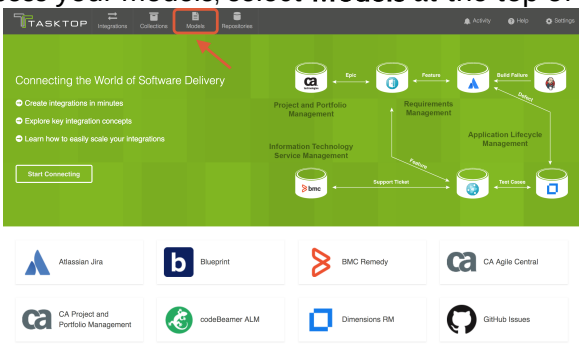
---

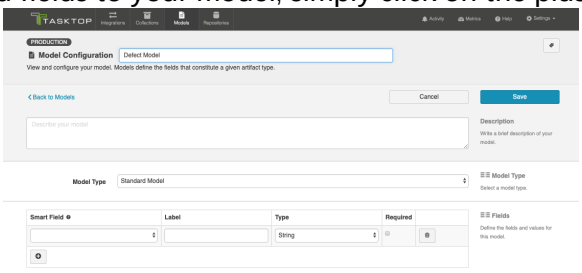
# Create or Reuse a Model

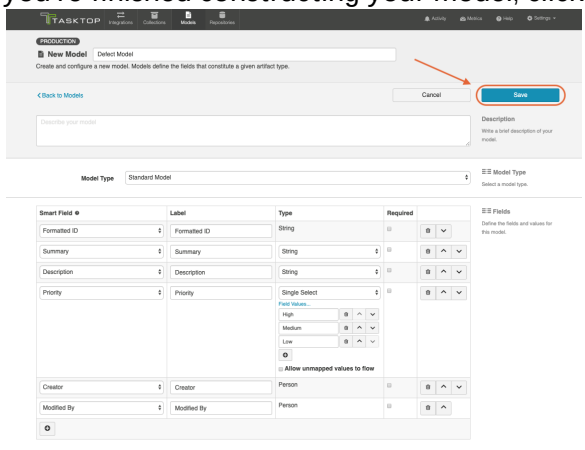
A **model** is a tool that makes the integration process scalable by defining the fields for each artifact type you would like to integrate.

Tasktop comes pre-packaged with several out-of-the-box models that are ready for you to use! If you'd like to use a pre-packaged model, you can skip this section and start creating your collections.

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

1. To access your models, select **Models** at the top of the screen.
   a. 

2. If you'd like to create a new custom model, click **+ New Model** at the top of the screen.
   a.
3. To add fields to your model, simply click on the plus sign at the bottom left of the model box.
   a. 

4. Once you're finished constructing your model, click **Save** and **Done** to save your model.
   a. 


For detailed instructions on creating or reusing models, click **here**.

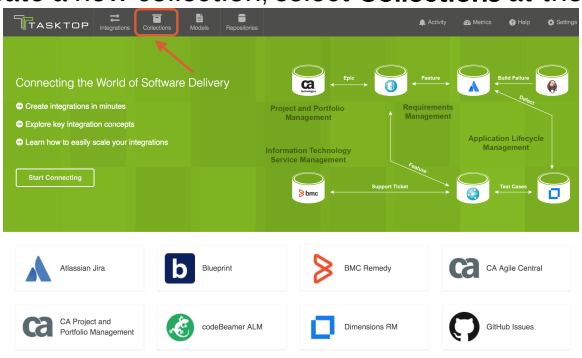
# Create your Collections

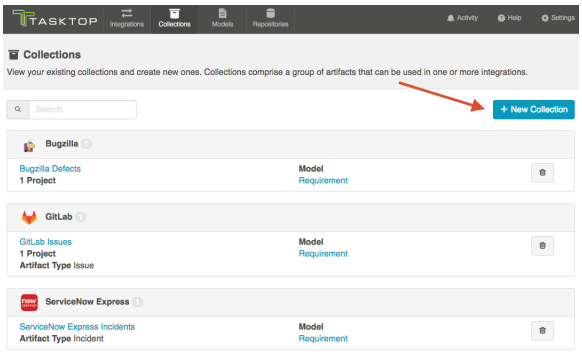
The next step to take when configuring an integration is to create your **collections**.

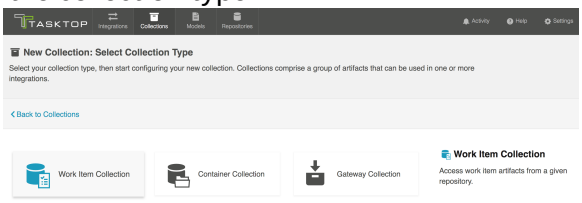
Your collections define which artifacts are eligible to flow as part of your integration.

You can create five types of collections: **Work Item (Repository)**, **Container**, **Work Item (Database)**, **Gate way**, **Outbound Only**.

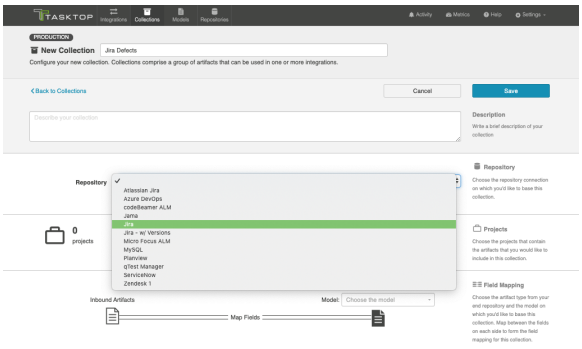
See **Tasktop Editions table** to determine if your edition supports all collection types.

1. To create a new collection, select **Collections** at the top of the screen.
   a. 

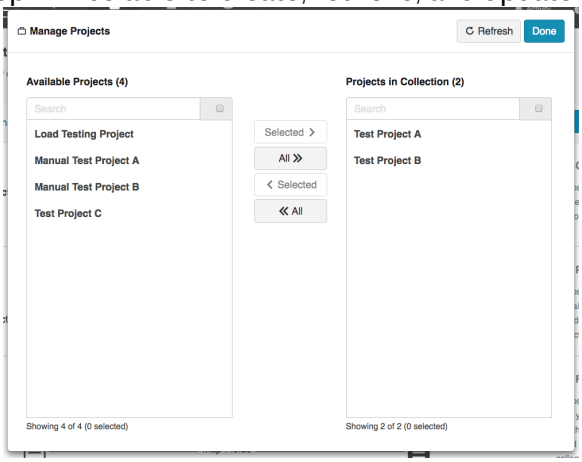
2. Click **+ New Collection**.
   a. 

3. Select the collection type.
   a. 

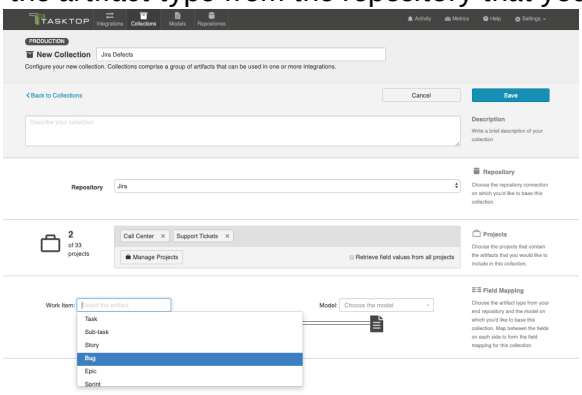
4. Enter a name for your collection and select the repository that you would like to connect. The collection will include artifacts from the repository you have selected.

a. 

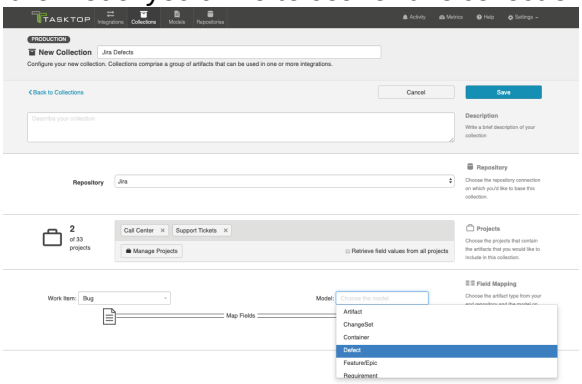
5. Add projects to your collection by selecting **Manage Projects**. These are the projects from which Tasktop will be able to create, retrieve, and update artifacts.

a. 


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

a. 

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

a. 

8. Click **Save** and **Done** to save your collection.

For detailed instructions on creating specific collection types, click **here**.

---

# Configure your Integration

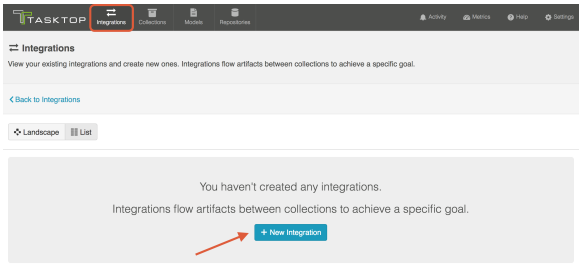
The last step is to configure your **integration**.

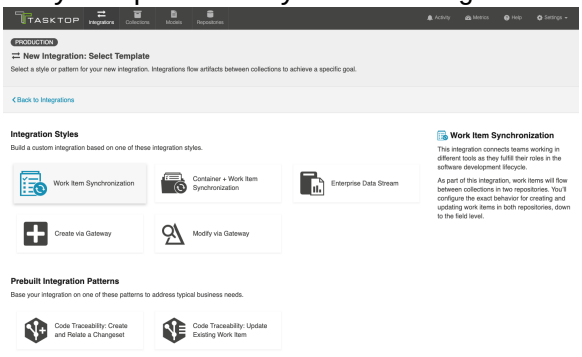
An integration is simply the flow of information between 2 or more tools.

Tasktop offers a variety of integration styles for your custom integration: **Work Item Synchronization**, **Container + Work Item Synchronization**, **Enterprise Data Stream**, **Create via Gateway**, **Modify via Gateway**.

Tasktop also offers prebuilt integration patterns for your integration: **Code Traceability: Create and Relate a Changeset**, **Code Traceability: Update an Existing Work Item**, **Test Synchronization**.

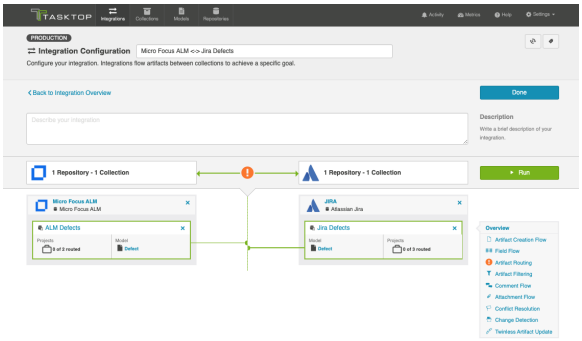
See **Tasktop Editions table** to determine if your edition supports all integration styles and patterns.

1. To configure a new integration, select **Integrations** at the top of the screen, then click **+ New Integration**.
   a. 

2. Select a style or pattern for your new integration.
   a. 

3. Name and describe your integration, and select your repositories and collections.

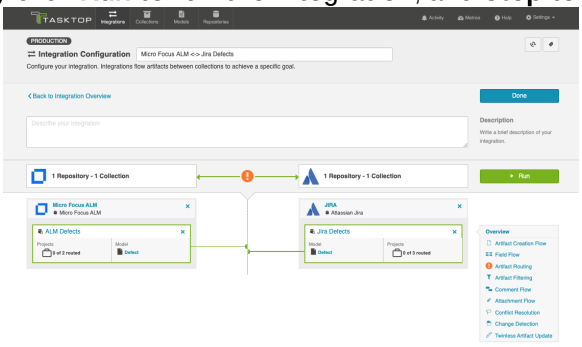
a.




# Running your Integration

Once you've completed your integration configuration, it's time to run your integration!

- Simply click **Run** to run the integration, and **Stop** to stop the integration.




For detailed instructions on configuring specific integration styles, click **here**.

For detailed instructions on configuring specific integration styles, click **here**.

# 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** |
|---|---|
| *Standard Repositories are available in all Editions.* | *Database Repositories 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.* |
| A **standard repository** refers to an external tool, such as Jira or ServiceNow.<br><br>These are software lifecycle tools that contain artifacts, such as defects or requirements. | A **database repository** refers to an external database, such as MySQL or Oracle.<br><br>Database repositories are used as part of the Enterprise Data Stream add-on. |
| Learn More | Learn More |

💡 **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 **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:

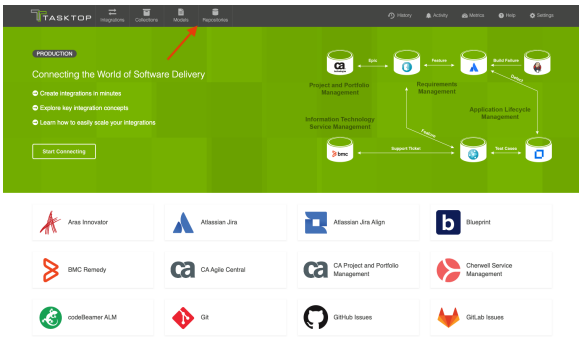
## Before You Begin

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.

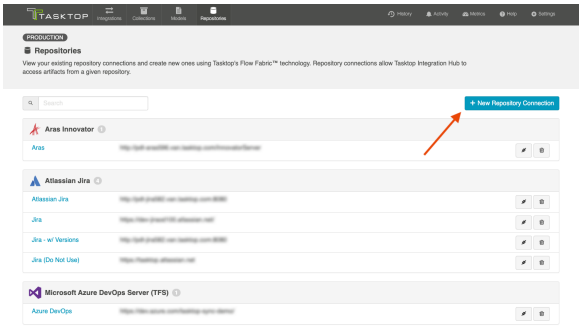
## Connecting to a Standard Repository

### Creating a New Connection

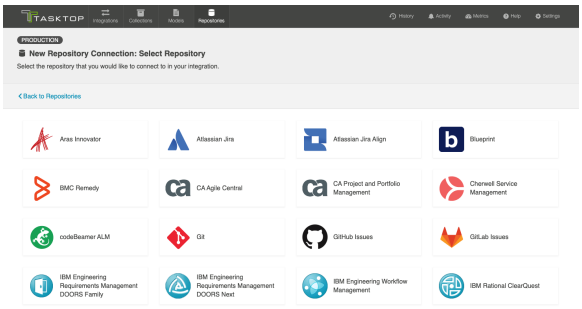
To create a repository connection, select **Repositories** at the top of the screen.




Click **+ New Repository Connection**.

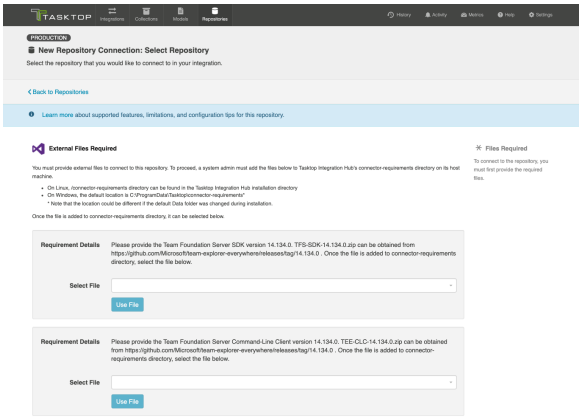



Select the logo of the repository you would like to connect to in your integration.




## 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**.

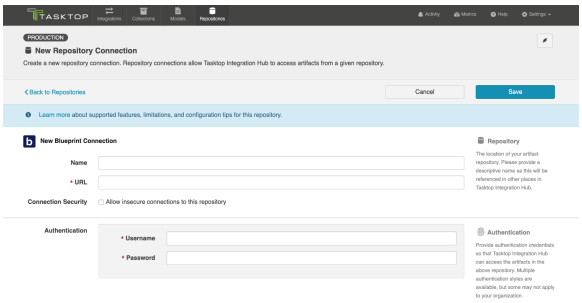
## New Repository Connection Screen

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.




## 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 Integration 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.

Standard Authentication

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.


SSO Authentication

If you connect to a repository utilizing CA SSO authentication, you can select one of the additional authentication options offered.

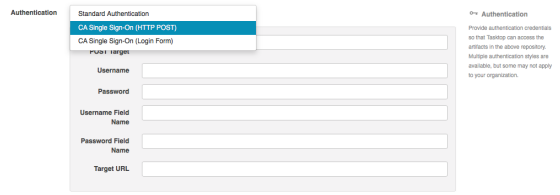
Tasktop currently supports the following SSO implementations:

- CA Siteminder/CA Single Sign-On (HTTP POST)
- CA Siteminder/CA Single Sign-On (Login Form)

- Script (HTTP cookies)
- X.509 Certificate

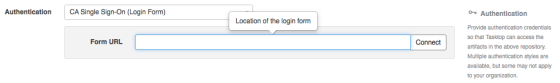
### HTTP POST

The **HTTP Post** option will generate the authentication form for you to fill in. Only the first 3 fields are required.




### Login Form

The **Single Sign-On (Login Form)** option will allow you to enter the URL for your SSO login form.




Once the URL is entered, Tasktop will auto-generate the fields that must be populated to connect to the repository.

### Script (HTTP cookies)

> ⚠ This method is not available for Tasktop Hub Cloud instances.

To use the **Script (HTTP cookies)** authentication method, a system administrator (a user with file system access to the machine that hosts Tasktop) must add the script(s) to the designated directory:

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

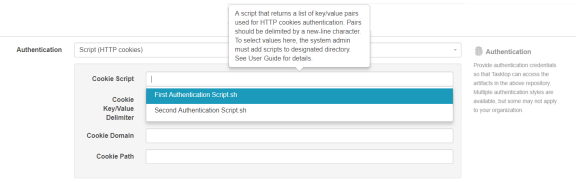
Once uploaded, select the script from the options available under the **Cookie Script** field. The script will be executed by the machine that hosts Tasktop. The script is stored in the Tasktop database, but is written to disk upon Tasktop startup and deleted from disk upon Tasktop shutdown.

Since Tasktop supports both Windows and Linux, please ensure that your script is able to be executed on the appropriate operating system: .bat for windows or shell script for Linux.

The **Cookie Script** will be executed and the standard out (and standard error) must read as a \n separated list of key/value pairs themselves separated by Cookie Key/Value Delimiter (default is **=**).

The **Cookie Domain** and **Cookie Path** arguments will then be used in the construction of a cookie for each of those key values pairs.

💡 **Note**: As Tasktop creates a copy of the script when the repository configuration is saved, changing the script in the directory will have no direct effect on existing repositories. For changes to a script to take effect, the user must go to the target repository connection and update the configuration.




### X.509 Certificate

To use the **X.509 Certificate** authentication method, select the X.509 Certificate to upload from your local machine. The certificate is stored in the Tasktop database, but is written to disk upon Tasktop startup and deleted from disk upon Tasktop shutdown.

Custom Authentication

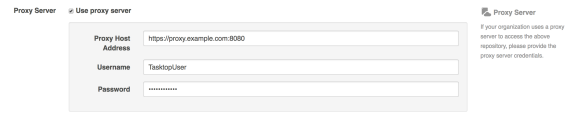
Some repositories allow for additional authentication methods. Please see our Connector Documentation for repository-specific information regarding authentication methods.

## 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 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 **user proxy server** box and fill in your proxy details in the **Proxy Server** section on the New Repository Screen:



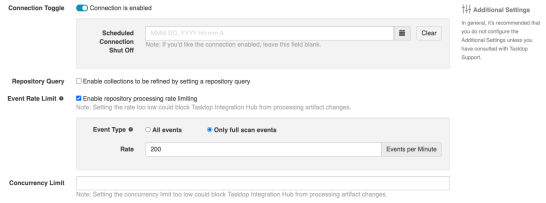

## Additional Settings

⚠️ **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




Connection Toggle

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.

Repository Query

If you plan to utilize a repository query, select the **Repository Query** checkbox.

⚠ **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. You can learn more about repository queries here.

Event Rate Limit

The **Event Rate Limit** can be used to mitigate scenarios where an external repository is temporarily receiving an excessive API call rate over short periods of time. It does this by limiting the number of events processed per minute by Tasktop for that repository. Events include Tasktop processes such as artifact retrieval, artifact update, and change detection queries. On average, an event consists of about 3-10+ API calls, but this is highly dependent on the specific repository.

When setting an Event Rate Limit, you can choose to limit:

- **All events**
- **Only full scan events**: If this is selected, only low priority events occurring during a full scan will be limited. High priority events, such as artifact updates, will continue without impact.

Note that the rate set is a maximum rate; Tasktop may process items at a lower rate depending on the event load.

Tasktop's default event rate limit is applied to full scan (observe) events only, at 200 events per minute.

⚠ **Note**: Caution should be used when setting this value. The ideal Event Rate Limit 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 low when there is a large number of projects

configured in your collections and a low Change Detection Polling Interval setting can potentially cause Tasktop to be unable to process artifact changes.

Concurrency Limit

The **Concurrency Limit** is set at the Repository level, and it limits how much work Tasktop can do in parallel in that repository. It does this by limiting the number of concurrent tasks where the connection is used. We recommend leaving this field blank/set to the default (having no specified limit).

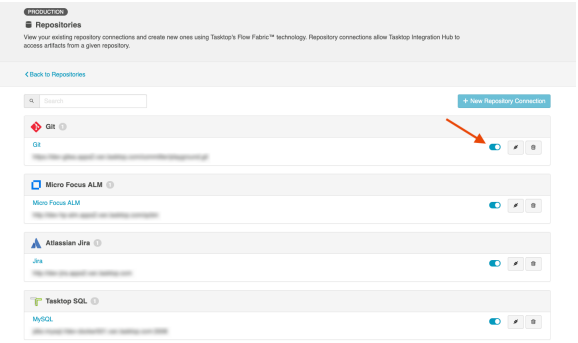
If you notice that Tasktop is placing too high a load on your repository, we recommend first modifying the Event Rate Limit. If that is insufficient, or if the source of the high server load is specifically due to having too many open connections on the external repository, the Concurrency Limit can be modified. We recommend starting with a value between 3-10 and engaging with support to determine an appropriate value for your unique environment.

⚠️ **Note**: Caution should be used when setting this value. The ideal Concurrency Limit 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 low when there is a large number of projects configured in your collections and a low Change Detection Polling Interval setting can potentially cause Tasktop to be unable to process artifact changes.

# Toggling your Repository Connection

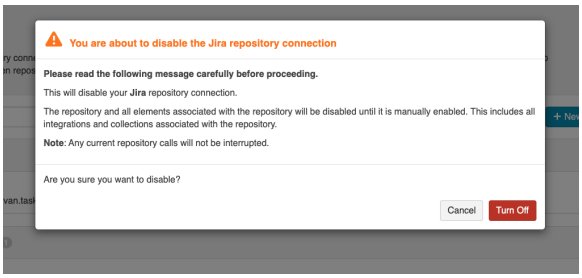
To enable or disable your repository connection, click the toggle on the **Repositories** screen.

💡 **Tip**: You can also toggle your repository connection on the **Repository Connection** screen.

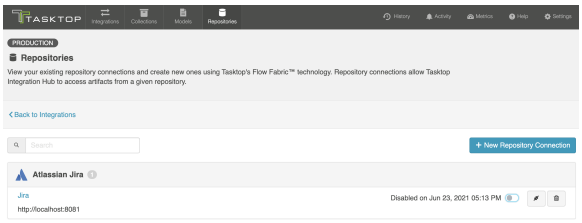



If you choose to disable your repository connection, a pop-up will appear confirming you'd like to proceed.

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

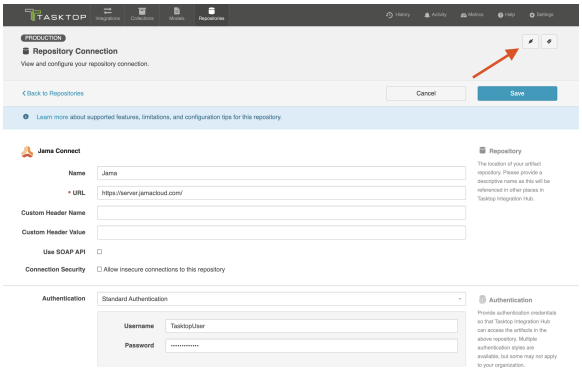


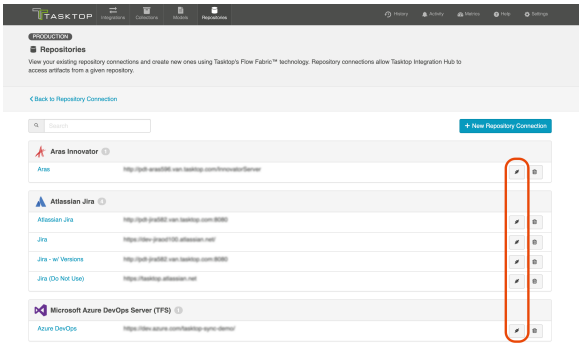

Once the connection has been disabled, the repository must be up and running to enable the connection again. If not, you'll see an error message at the bottom of the screen.



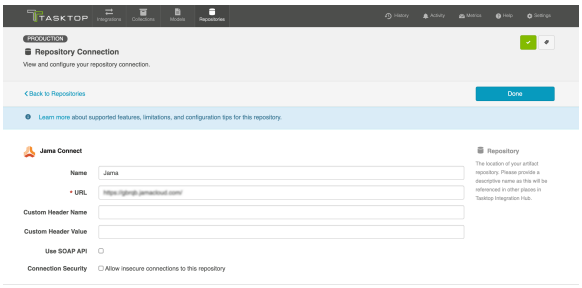

# Testing your Repository Connection

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

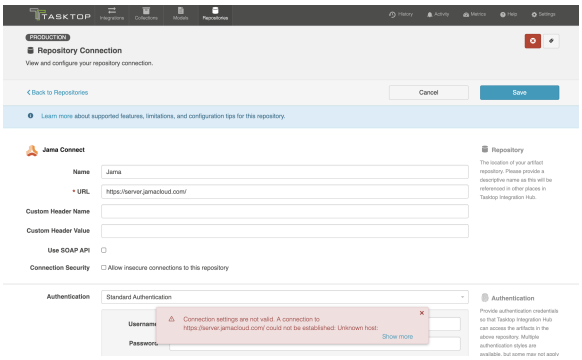


You will see a success or failure message to confirm whether Tasktop was able to connect to your repository.

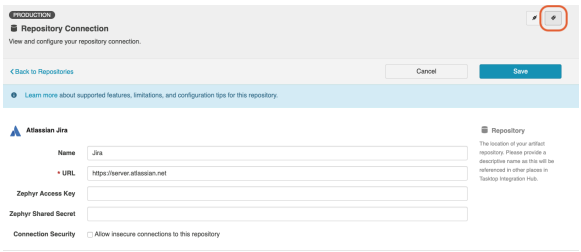


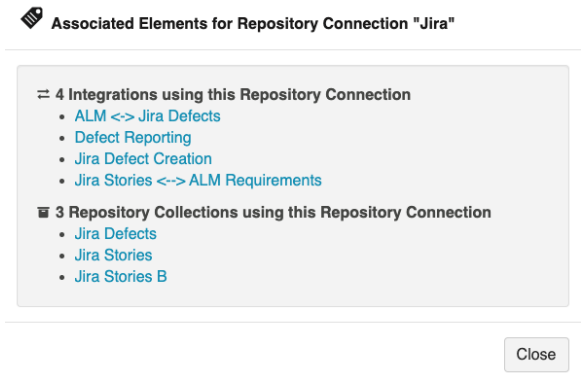

When your repository fails to connect, you will also see an error message at the bottom of the screen with additional details on the source of the failure.




# 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.

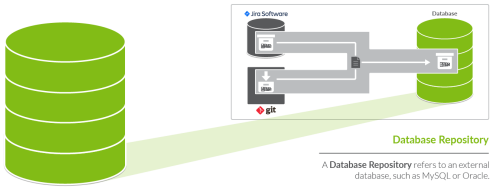



Associated Elements for Repository Connection "Jira"
4 Integrations using this Repository Connection
ALM <-> Jira Defects
Defect Reporting
Jira Defect Creation
Jira Stories <--> ALM Requirements
3 Repository Collections using this Repository Connection
Jira Defects
Jira Stories
Jira Stories B
Close

# 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 Tasktop 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

- 9.6 - 13

Extended Support

- N/A

💡 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

- 18c
- 19c

Extended Support

- N/A

### 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

For PostgreSQL, please refer to PostgreSQL documentation for more information.

### Location

- Example Format: *jdbc:postgresql://hostServerName:postgreSqlServerPort/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;databasename=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**)

💡 **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

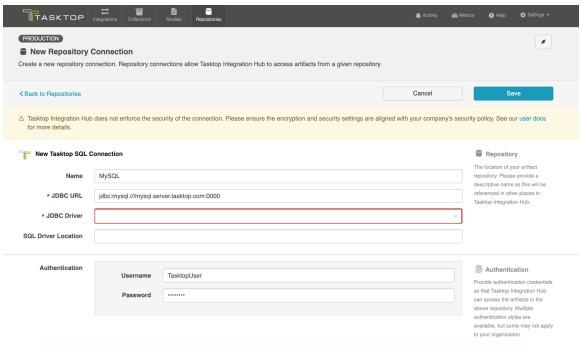
The SQL Driver Location should reference the directory containing the `ojdbc6.jar` file. The `ojdbc6.jar` file should be the only file in that directory, or you may end up with errors upon configuring your collection. Note that it is best if the Oracle JDBC driver that is used matches the version of the Oracle server that you are connecting to.

### PostgreSQL

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

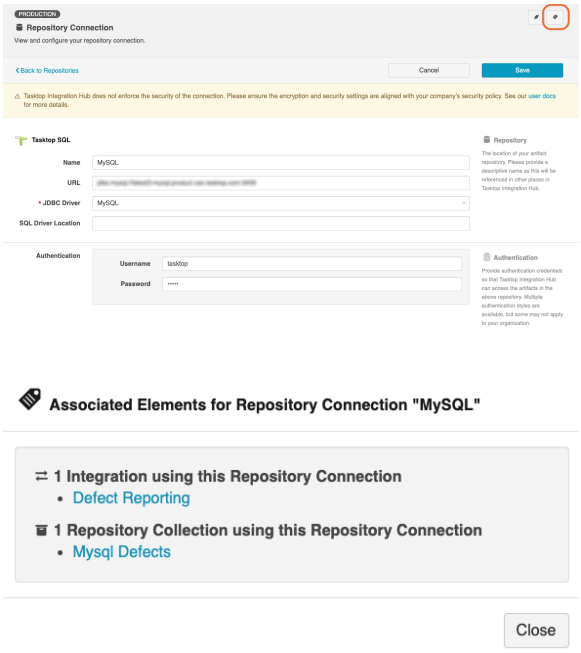
# Step 3: Connect to your Database

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.




# Viewing Associated Configuration Elements

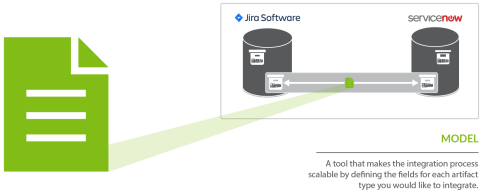
To view associated configuration elements (such as collections or integrations that utilize the repository connection you are viewing), click the **Associated Elements** tag in the upper right corner of the screen.


PRODUCTION
Repository Connection
View and configure your repository connection.
Back to Repositories
Cancel
Save
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
MySQL
URL
* JDBC Driver
MySQL
SQL Driver Location
Repository
The location of your artifact repository. Please provide a descriptive name as this will be referenced in other places in Tasktop Integration Hub.
Authentication
Username
tasktop
Password
Authentication
Provide authentication credentials so that Tasktop Integration Hub can access the artifacts in the above repository. Multiple authentication styles are available, but some may not apply to your organization.


Associated Elements for Repository Connection "MySQL"
1 Integration using this Repository Connection
Defect Reporting
1 Repository Collection using this Repository Connection
Mysql Defects
Close

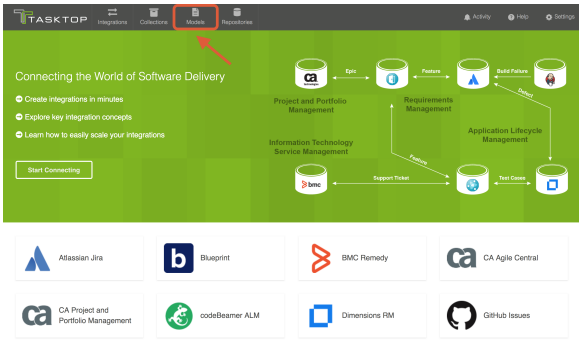
# Step 2: Create or Reuse a Model

## 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.

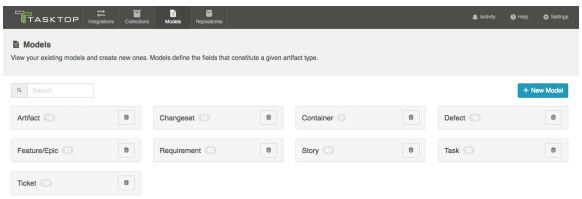
To access your models, click on the **Models** button at the top of the screen.



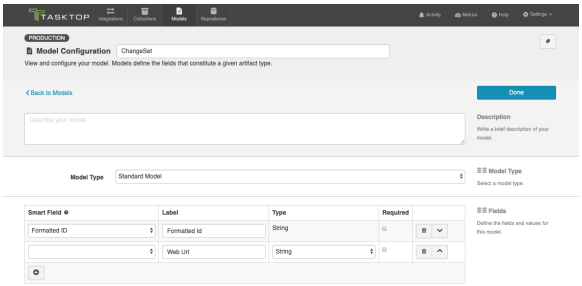

## Out of the Box Models

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.

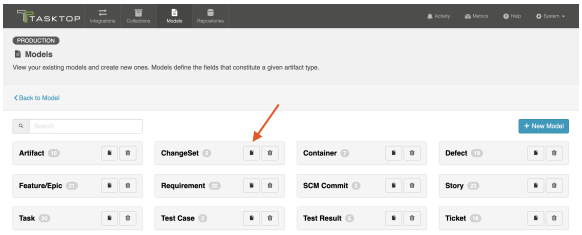



To view a model, simply click on its title. You will be brought to the Model Configuration screen, which will show the fields included in that model.




# Copying a Model

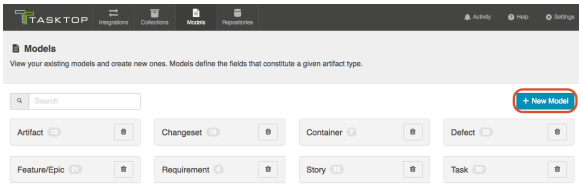
There may be times when you wish to copy a model and then modify certain fields on it for use in a different integration. To copy a model, click the **copy** button from the Models screen. The copied Model will be named "<Original Model name> (copy)".




# 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.

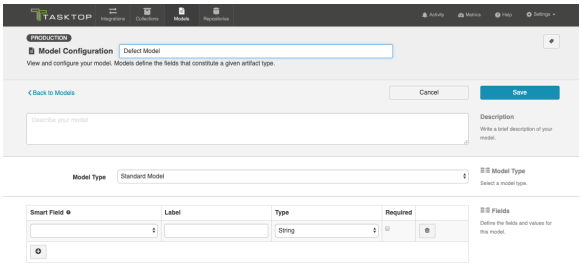



# Model Type

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

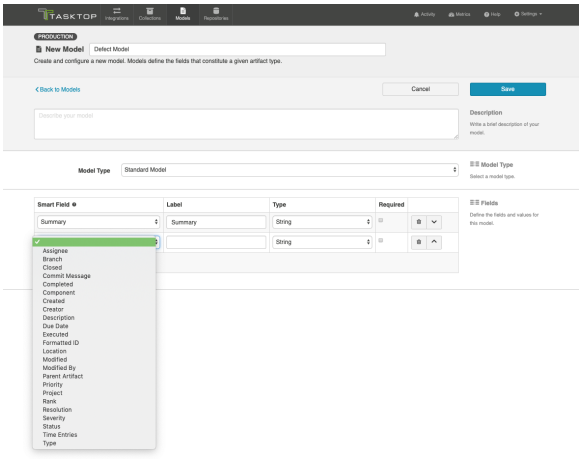
For each field you add to your model, you have the option of identifying its corresponding smart field type. *Smart fields* are a set of fields commonly available in the connectors for all of the repositories Tasktop connects to. By designating a smart field to your model field, Tasktop will be able 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.

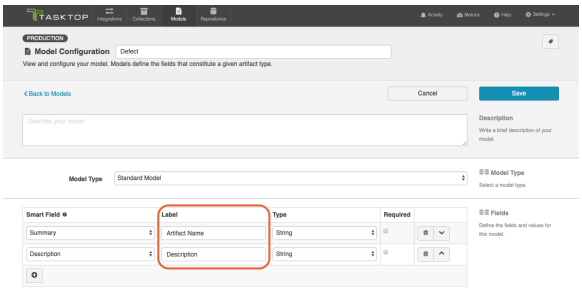
Some examples of smart fields are:

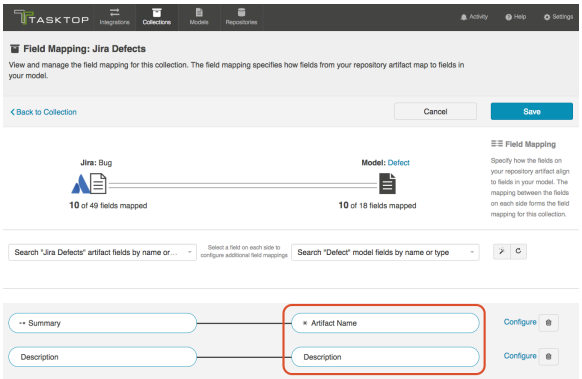
- **Formatted ID**: the human-readable ID of an artifact
- **Location**: the field that holds the URL of an artifact
- **Modified**: a date-time field showing when changes were last made to an artifact




## 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.

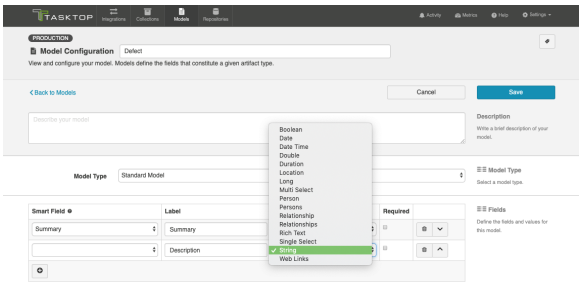



# Field Type

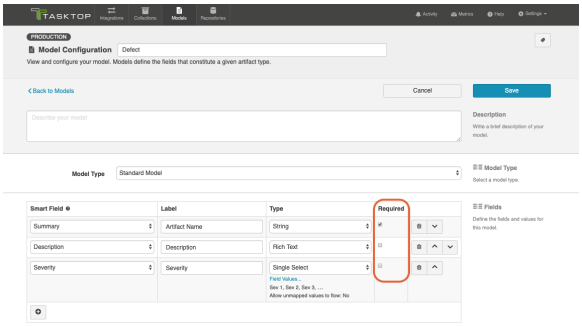
Tasktop supports a number of field types, such as *string, multi-select, relationship,* and more, for use in your model. Identify the field type that most closely aligns with the type of information you expect to flow through this model field.

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



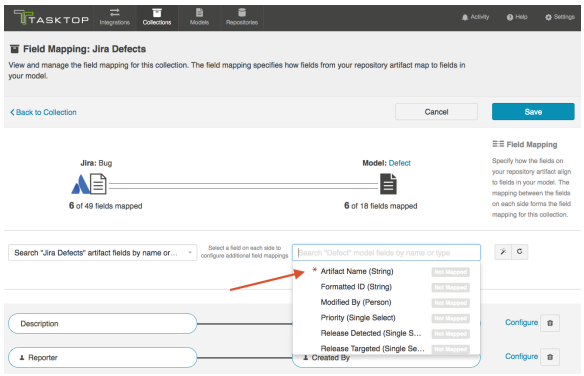

# Required Designation

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



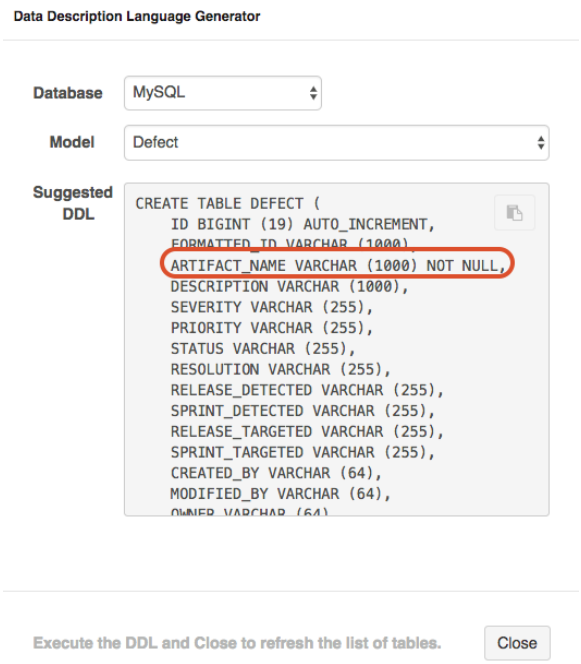

Marking a field as required has implications for all collection types:

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



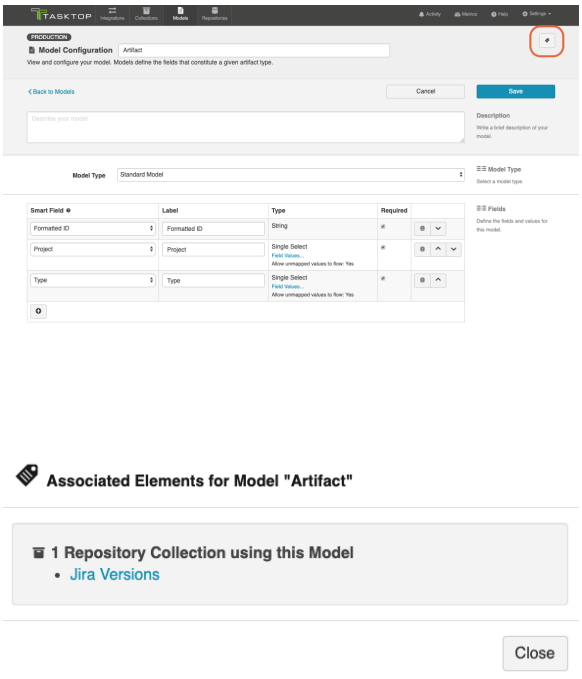

- For gateway collections, you will need to pass in a value in the payload for any required field in order for Tasktop to accept the payload.

- 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.




# 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.




# 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

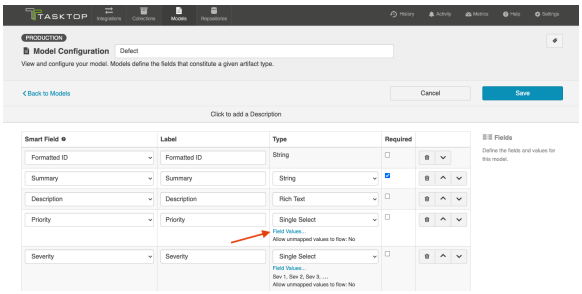
## Fields that Require Additional Configuration

### Single-Select and Multi-Select

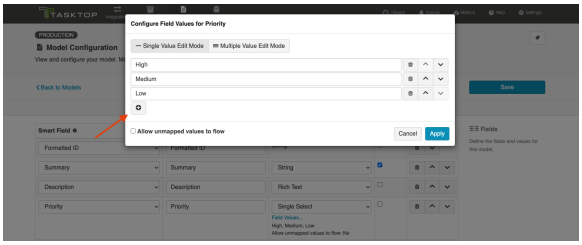
Single-selects and multi-selects fields refer to fields in which the user selects one or many options from a list of values. These fields could refer to drop down menus, checkboxes, or radio buttons within the end repository, to name a few examples.

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

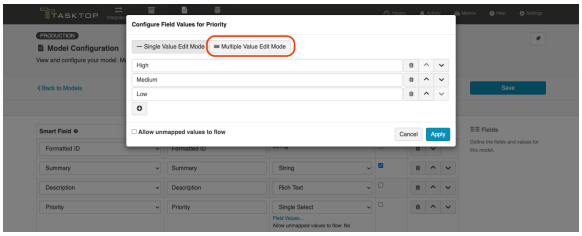
First, click the **Field Values** link to add values to your model. These will be the available field values that you will then map to fields within each end repository.



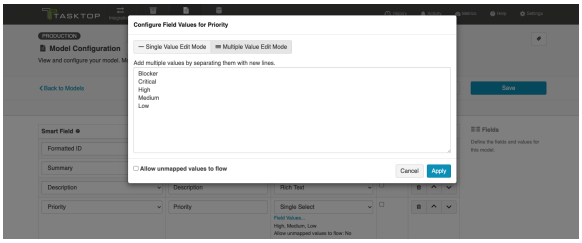
After clicking the link, a pop-up will appear prompting you to configure your field values. If you'd like to add a single field value to your model, you can use the **+** button to do so.



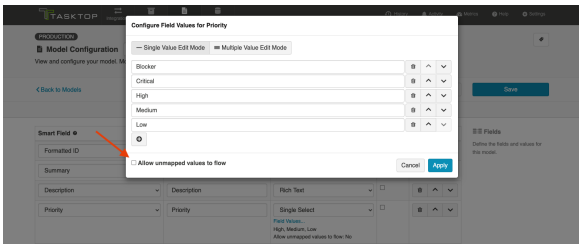
If you'd like to add multiple values at once, select **Multiple Value Edit Mode**.



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

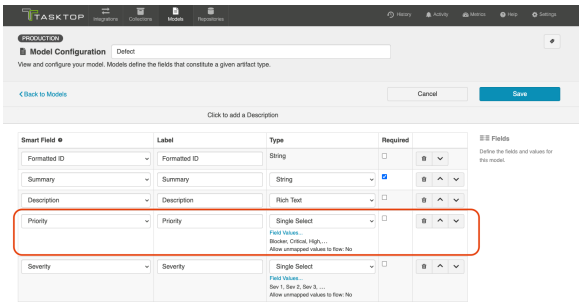


Next, decide whether or not you'd like to allow unmapped values to flow by checking this box.



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.

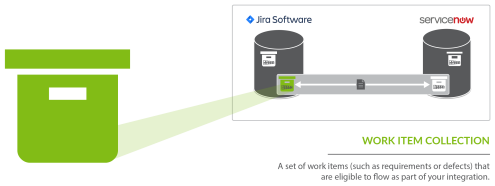
You can create five types of collections:

|  **Work Item Collection (Repository)** |  Container Collection (Repository) |  **Work Item Collection (Database)** |  **Gateway Collection** |  **Outbound Only Collection** |
|---|---|---|---|---|
| *Work Item Collections (Repository) are available in all Editions.* | *Container Collections (Repository) are available in all Editions.* | *Work Item Collections (Database) 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.* | *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.* | *Outbound Only Collections are only available in editions that have access to the Git repository.* |
| A **work item collection (repository)** contains work items, such as defects or requirements, | A **container collection** contains containers, such as folders or modules, from | A **work item (database) collection** connects to a database, such as MySQL or Oracle. | A **gateway collection** contains artifacts sent via an in-bound webhook, from an external tool. | An **outbound only collection** contains artifacts like code commits or changesets, which you may want to flow out of your repository, but |

| | | | | |
|---|---|---|---|---|
| from repositories, such as Jira or ServiceNow. | repositories such as DOORS Next Generation or Jama. | | | which would not receive updates into y our repository. |
| Learn More | Learn More | Learn More | Learn More | Learn More |

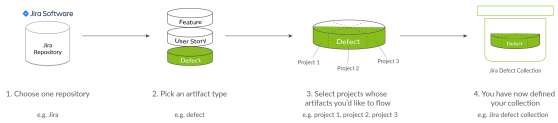
# Work Item Collection (Repository)

## 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.

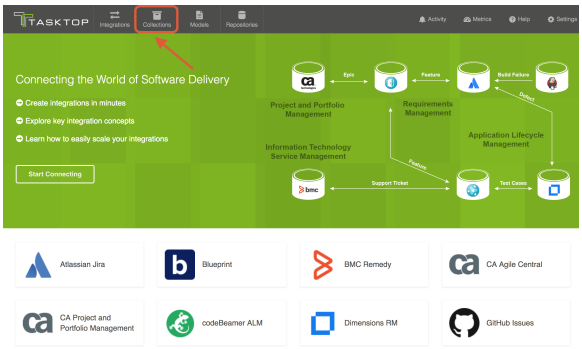
# Video Tutorial

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

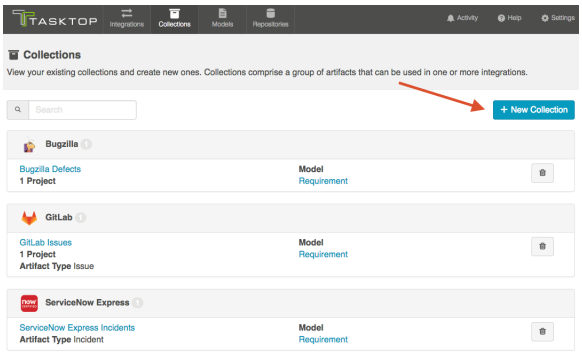
# Creating a Work Item (Repository) Collection

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

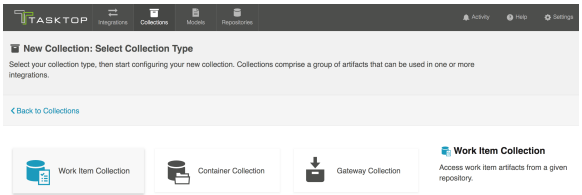
Select **Collections** at the top of the screen.



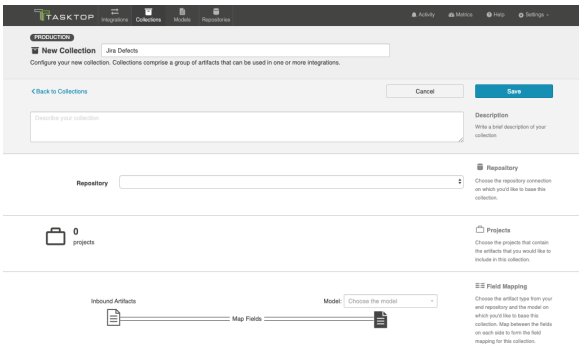

Click **New Collection**.



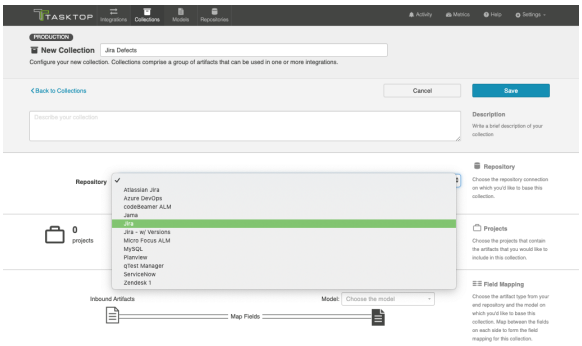

Select **Work Item Collection** as the collection type.




Name and describe your collection.

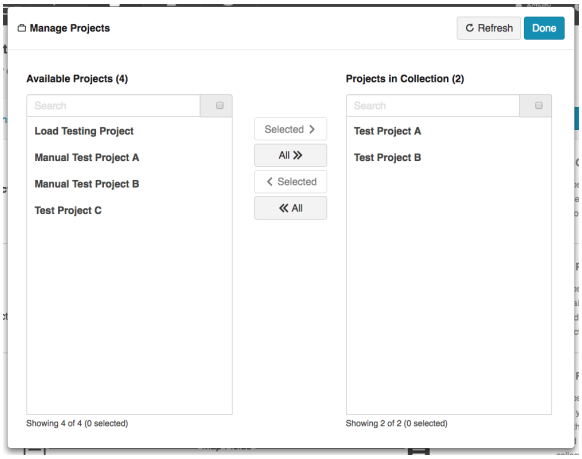



Select the repository that you would like to connect to. The collection will include artifacts from the repository you have selected.

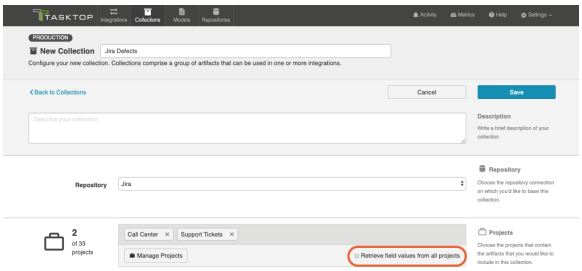



Add projects to your collection by selecting **Manage Projects**. These are the projects from which Tasktop will be able to create, retrieve, and update artifacts.

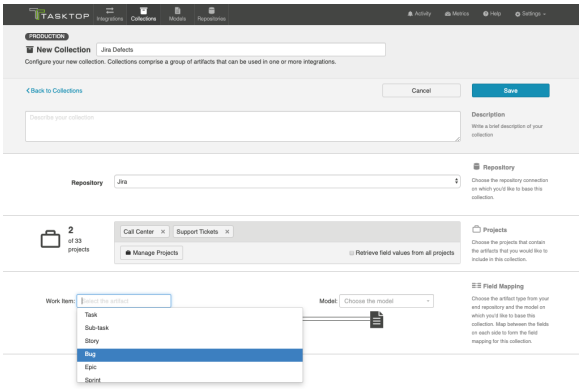
💡 **Note**: In some cases, the word **Project** is used loosely. You may be selecting workspaces or some other organizational structure, depending on the repository you've connected to. You can review our C onnector Docs to see which containers are supported for each repository.



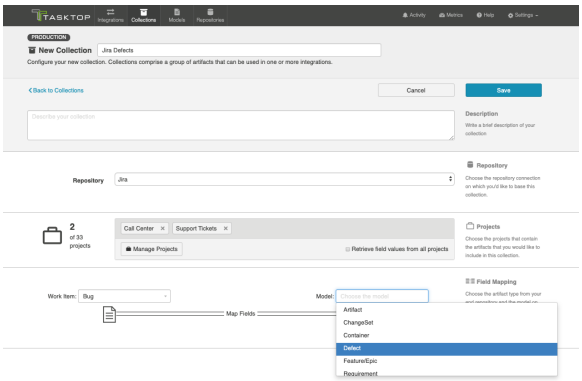

By default, Tasktop retrieves field values from one sample project for mapping. In rare cases where values vary between projects, check the **Retrieve field values from all projects** box on the Collection configuration screen to retrieve all possible values. Be aware that retrieving values from all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




Select the artifact type from the repository that you would like to include in this collection. Remember, a single collection can only contain artifacts of a single type.

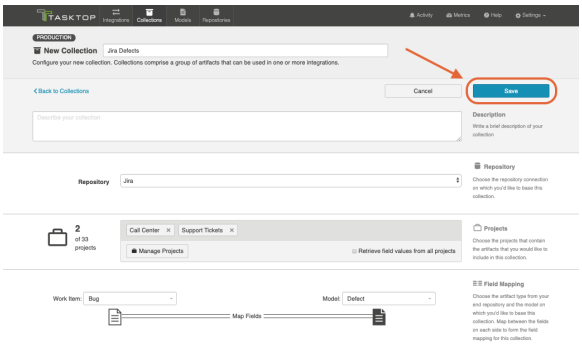



Select the model you'd like to use for this collection.

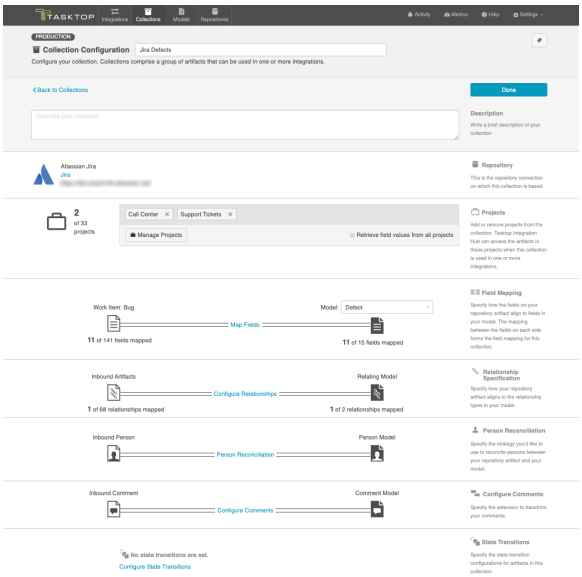



⚠ **Note**: The projects included in your collection must contain at least one artifact of the type selected. For example, in the image above, there must be at least one bug in Test Project A in Jira in order for your collection to save.

Click **Save**.




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.

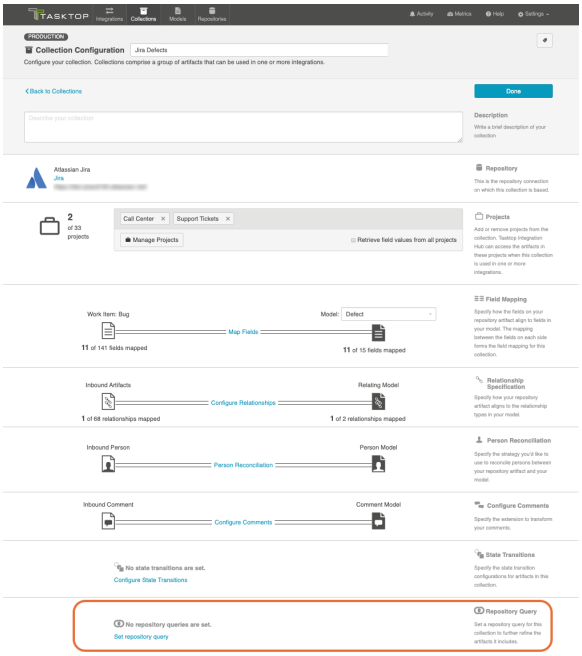
## State Transitions

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.

## 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.

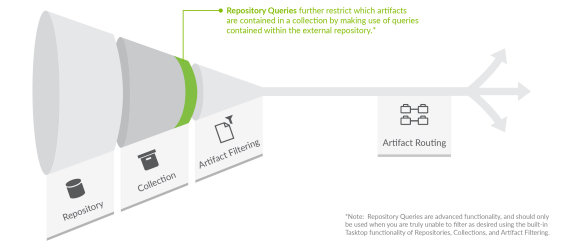
When configuring your integration, you have several options available to refine which artifacts are eligible to flow.

- 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.

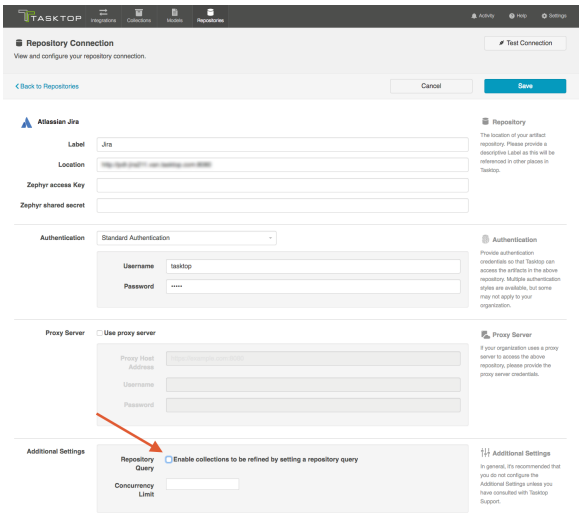
⚠ **Note**: When setting a repository query, we recommend only including artifacts of the same type and in the same project(s) as the collection. If a repository query contains many artifacts not included in the collection, performance may be impacted.

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.

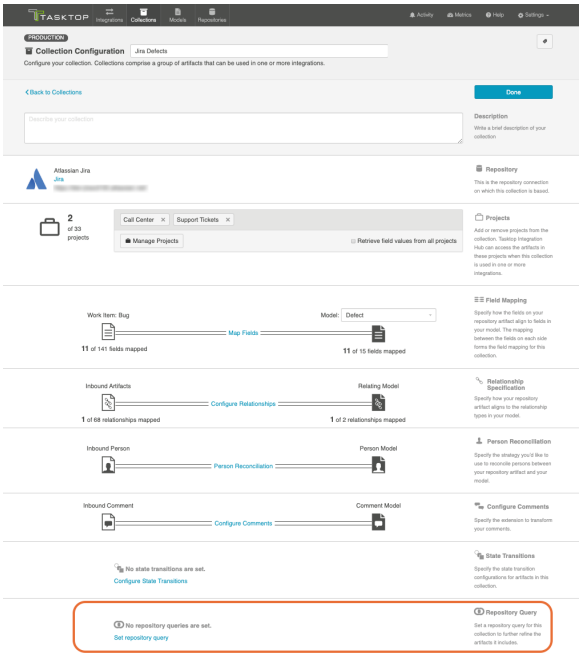
In rare cases, however, you may find that the best option to restrict the artifacts eligible to flow is by setting a query within the repository itself.



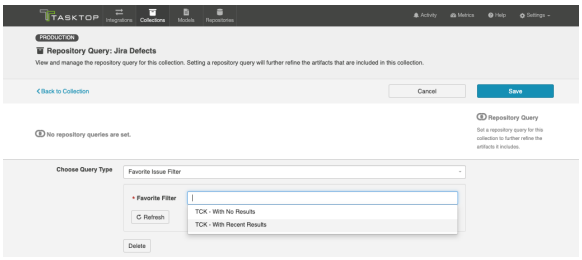

If you plan to utilize repository queries, check the box next to **Enable collections to be refined by setting a repository query**, on the Repository Connection screen.



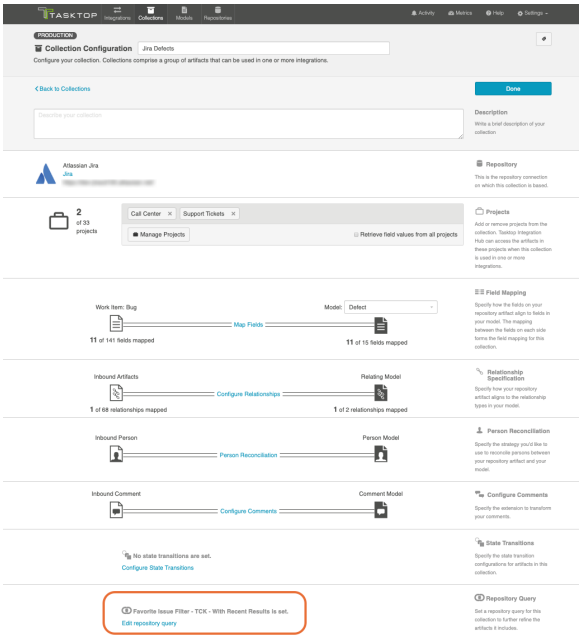

Once this is selected, you will be able to select a repository query at the Collection level for any collections utilizing this repository.




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



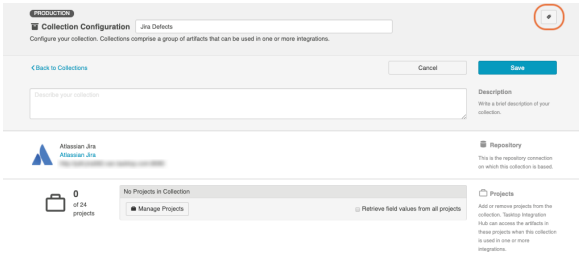

You will then see the selected repository query on the Collection Configuration screen:

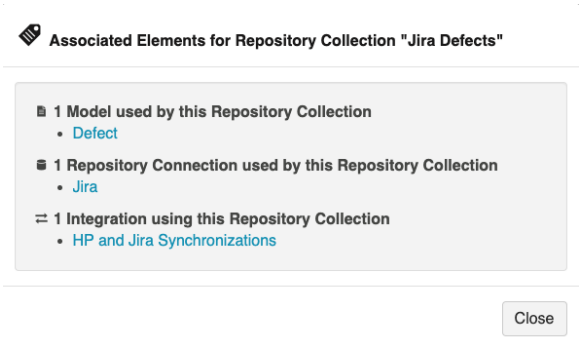



💡 **Note**: Remember, applying a repository query to a collection will only further refine the artifacts included in that collection. If you select a query that encompasses artifacts in projects not in your collection, these artifacts will not be added to the collection unless you also add those projects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

# 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

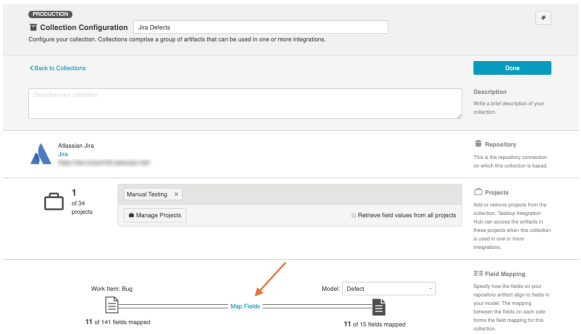
# Field Mapping

## 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.

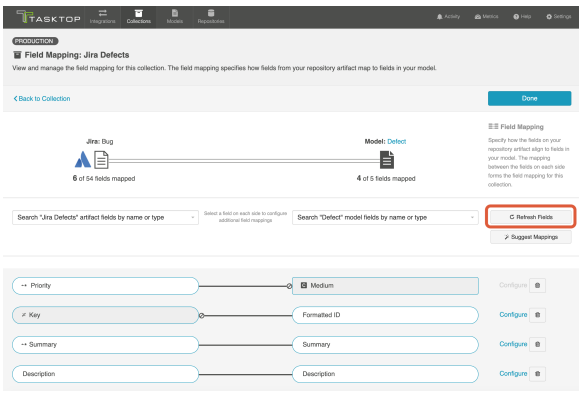
## Mapping Fields

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

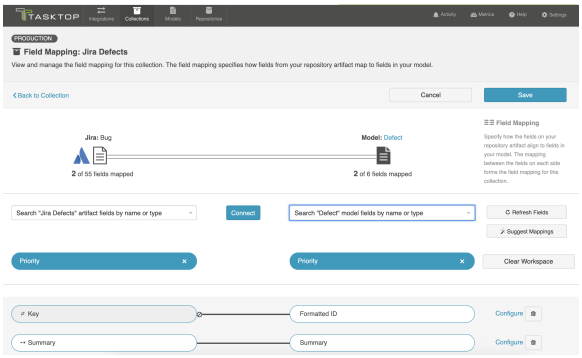



Clicking this link will take you to a drill in page where you can specify how the fields in your model will map to the fields available on the artifact within your repository. Tasktop will auto-map fields when possible based on the names of fields and the smart field designations that have been set in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



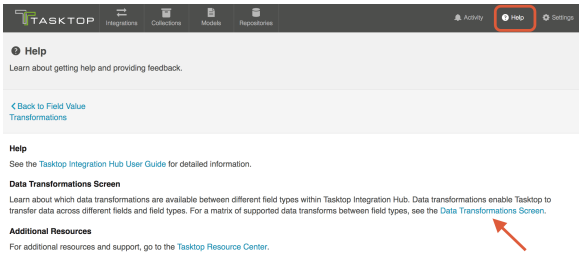

You can map additional fields by using the two drop down boxes:



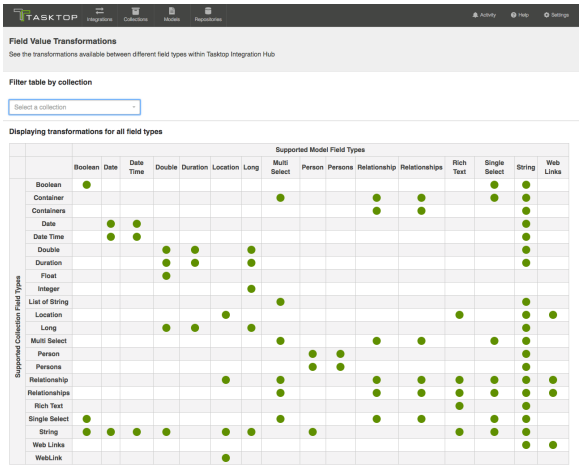

# Transforms

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.

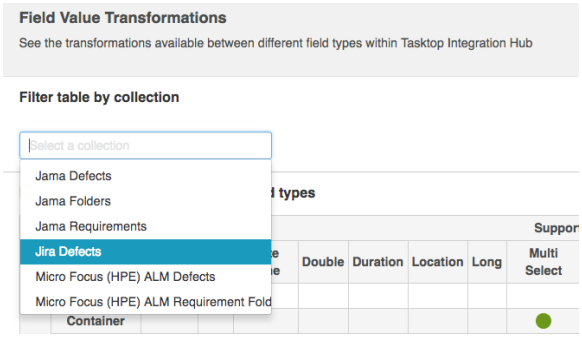
You can see a table of the available transforms by clicking the **Data Transformations Screen** link on the Help page.

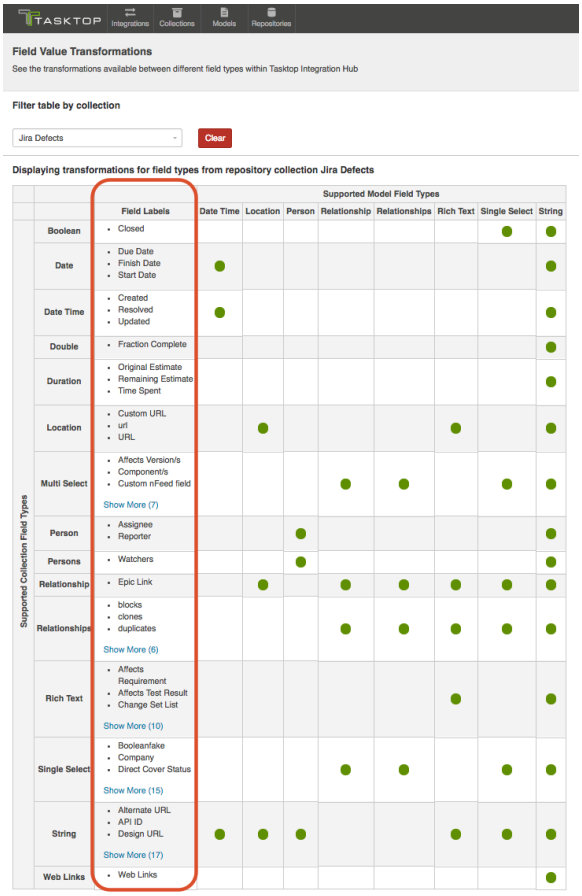



This will lead you to the Field Value Transformations screen. Here, you can see which collection-to-model field type transformations are available.

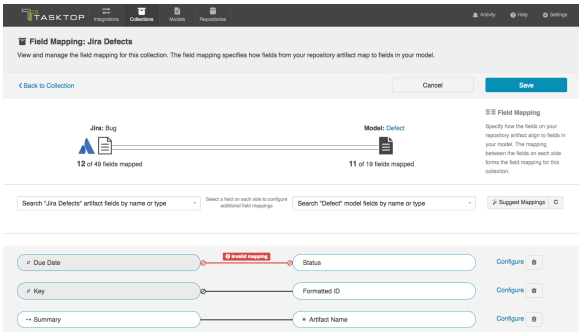



You can even filter by Collection to see the specific field labels and field types for that collection:

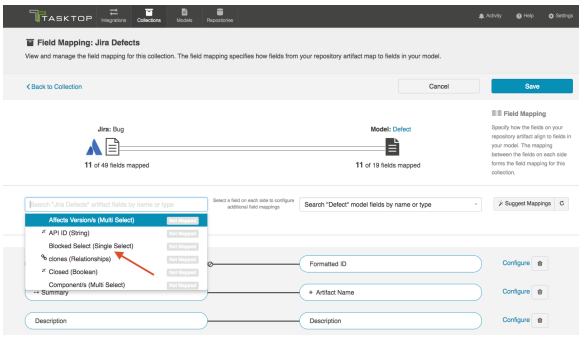



On the Field Mapping screen, if you attempt to map fields that do not have a valid transform between one another (for example, if you map 'due date,' a date field, to 'status,' a single-select field), you will get an 'invalid mapping' warning, and the mapping will not be saved.



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 |
|---|---|
| C | A constant value will be sent. Note that:<br>• 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 *other* collection will receive this constant value for the field in question.<br>• 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 *this* collection will receive the constant value for the field in question. |
| | A state transition will be utilized. Note that:<br>• If the icon is on the side of the collection, this means that a state transition graph is being utilized.<br>• If the icon is on the side of the model, this means that a state transition extension is being utilized. |
| | 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.

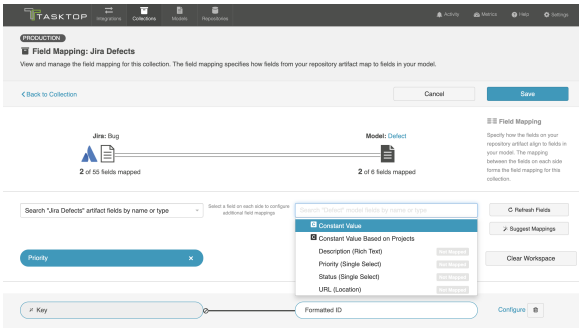
Constant values can be set for the following field types:

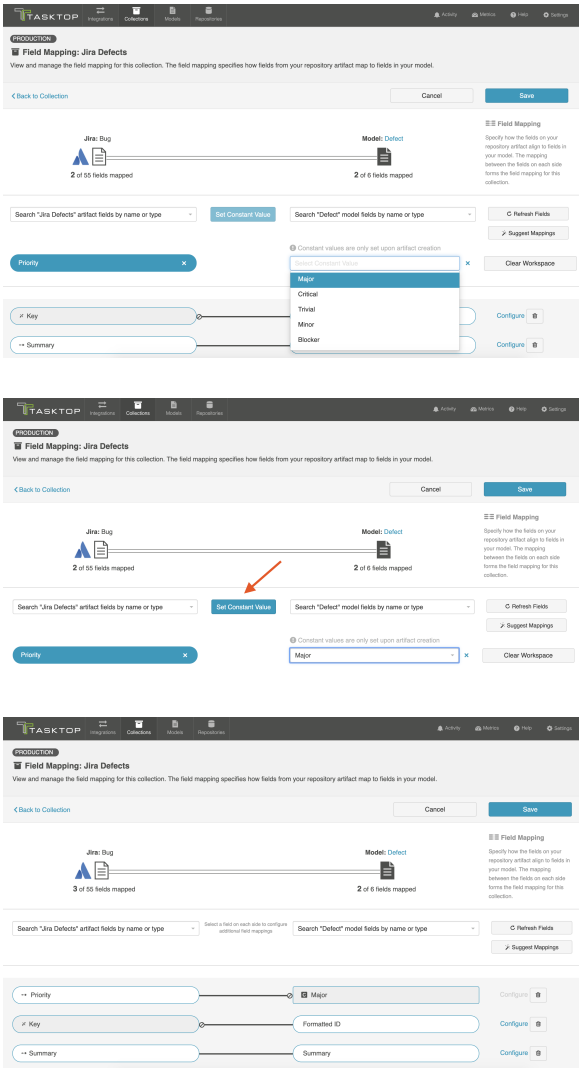
- Boolean
- Date/DateTime
- Double
- Location
- Long
- Multi-Select
- Person
- Rich Text
- Single-Select
- String

## 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

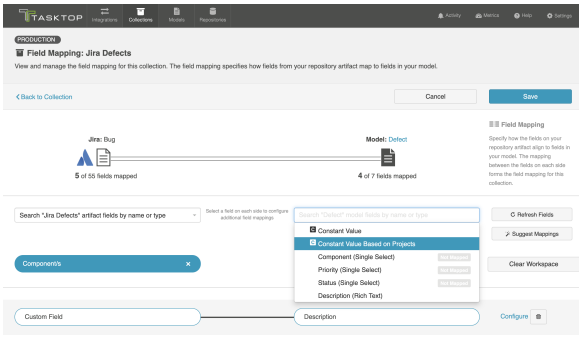
If desired, you can also set constant values per project.

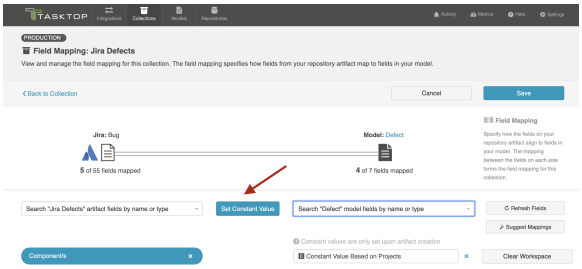
You may wish to set a constant value based on project in the following scenarios:

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

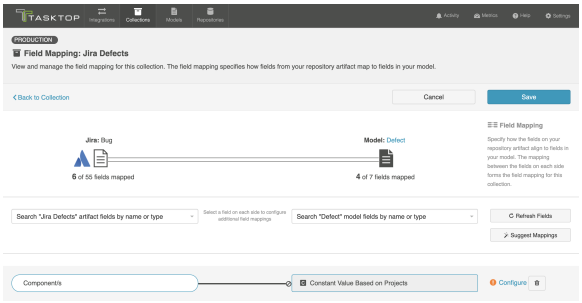
- If the values for a single-select field vary across projects

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

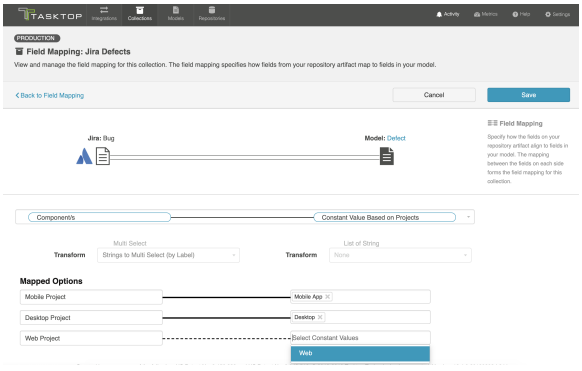



Once selected, you will see an orange exclamation point appear next to the 'Configure' link:

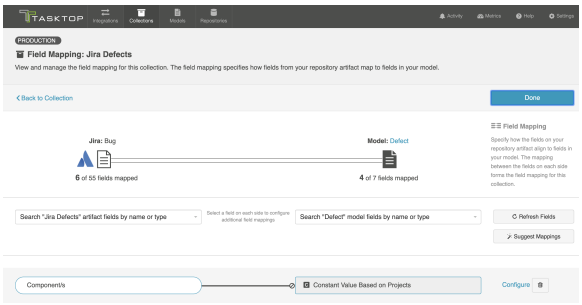



Click **Configure** to get to the Field Configuration Screen. On this screen, you will be able to set a distinct constant value for each project in your collection:




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** app

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




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

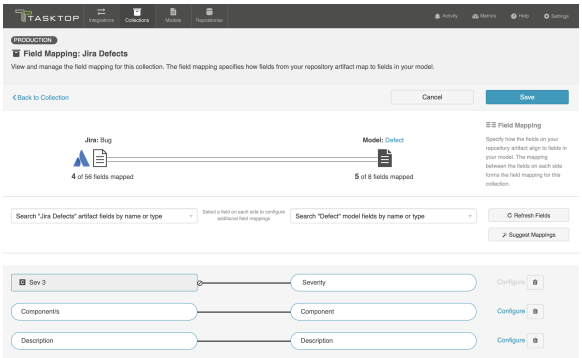
- The pill will be rectangular and grey: this denotes that a constant value has been set.
- The Constant Value icon will be displayed inside the pill.
- The 'prohibited' icon will appear next to the pill. This indicates that no values can be sent to the Constant Value field. The constant value is essentially 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.



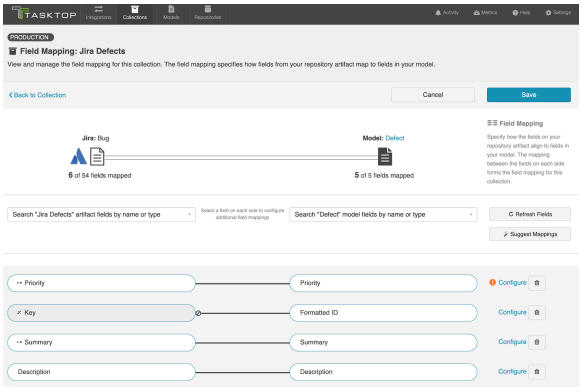

Once the constant value is set, you will notice a couple of things:

- 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.

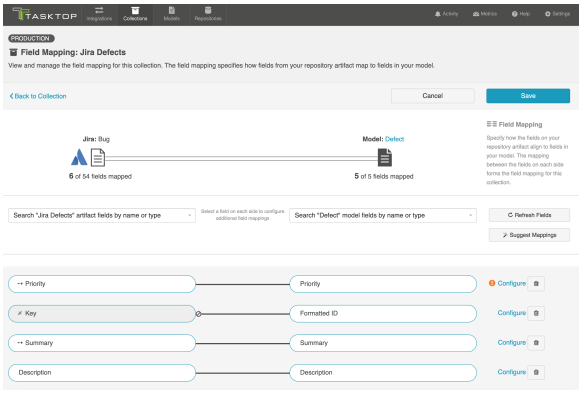



To review and update an individual field's configuration, click the **Configure** link to its right. You can learn more about Field Configuration on the Field Configuration page of our User Guide.

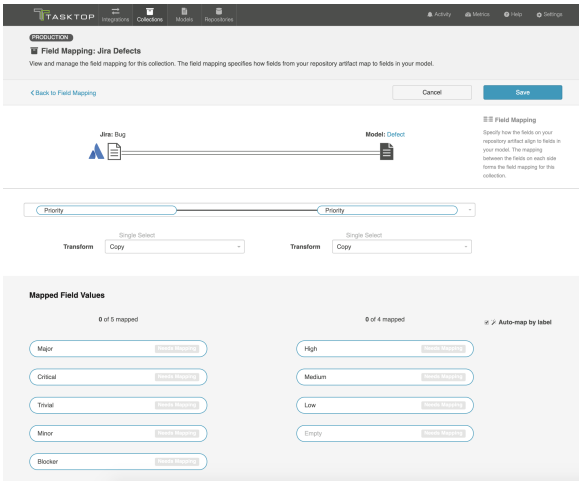
# Field Configuration

## 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:

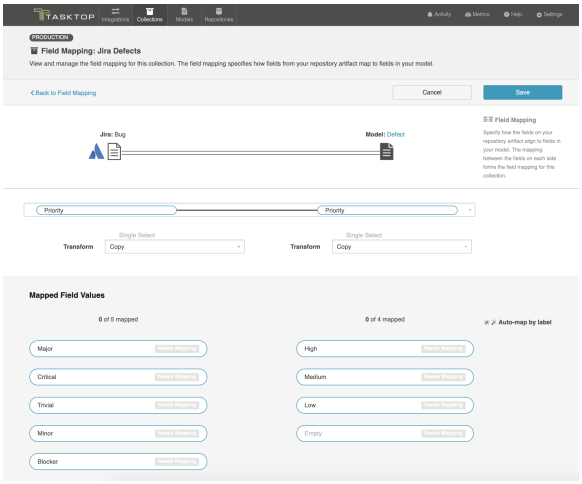



## Transforms

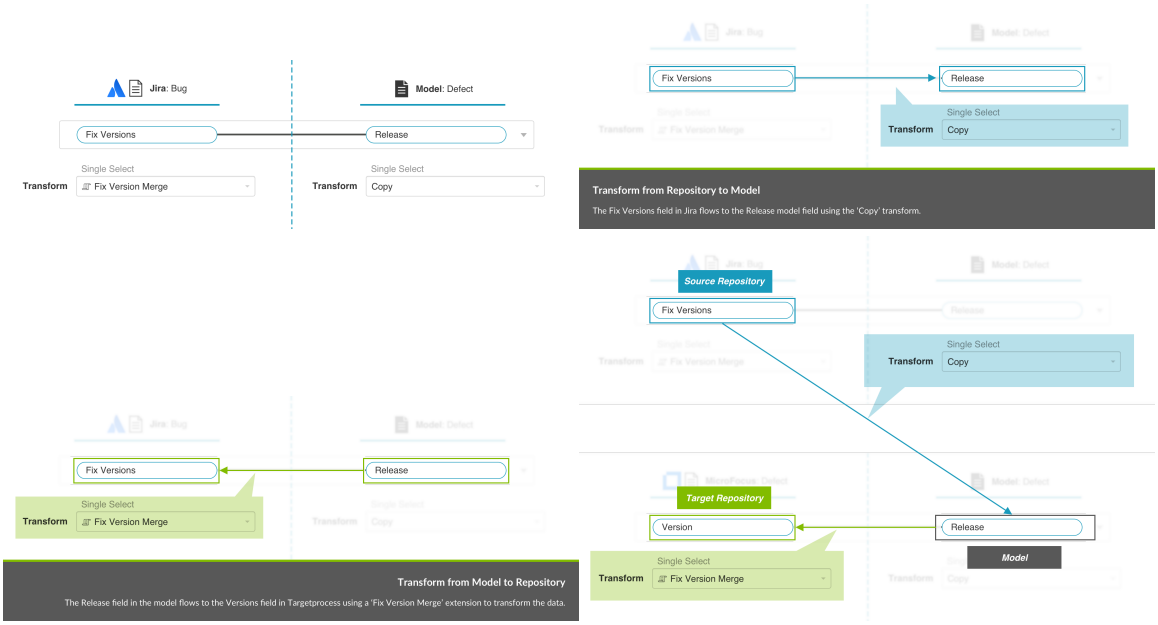
The Field Configuration screen is where you can configure your transforms and value mappings.

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.



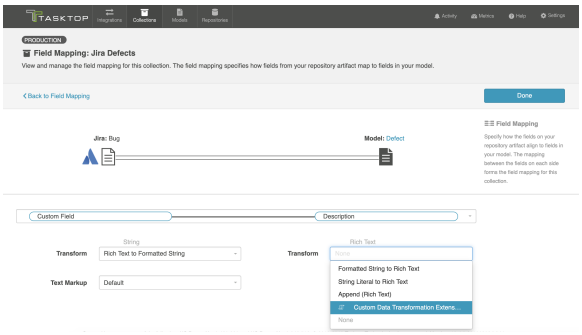

The transform on the left will impact how data flows into your repository (from your model), and the transform on the right will impact how data flows into your model (from your repository).




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 over-write 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 'bold,' 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.

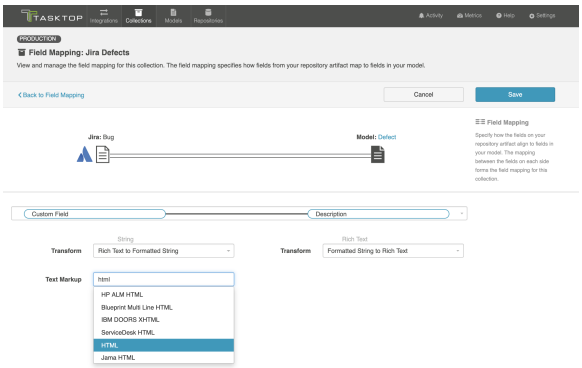
# Rich Text (Text Markup)

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.

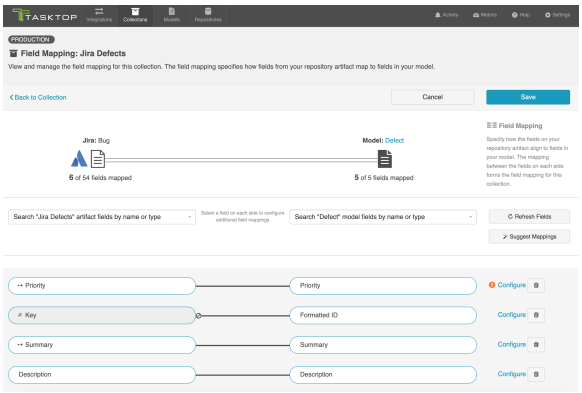
In cases like this, you can customize the desired text markup language. You'll see a wide range of text markup options available on this screen.  Search for the one you need and select it here. The language selected will impact both how data flows *into* and *out of* the collection for that specific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



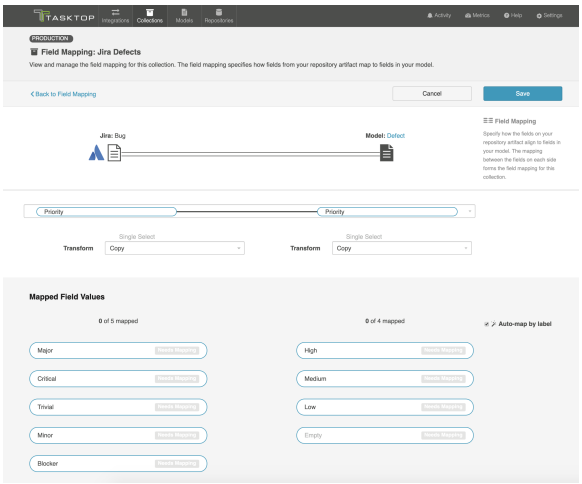

# Single- and Multi-Select Fields

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.



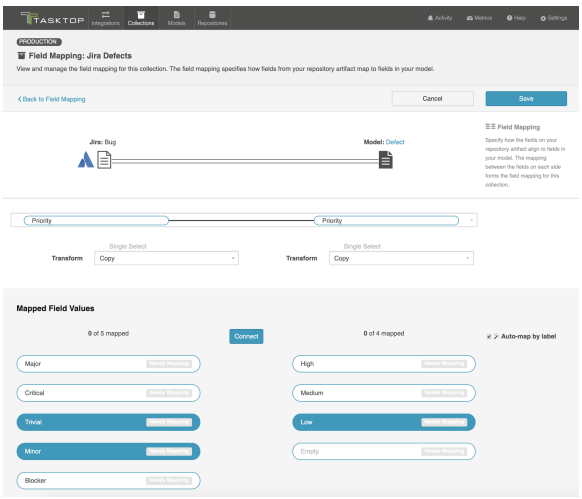
# Transforms for Single- and Multi-Selects

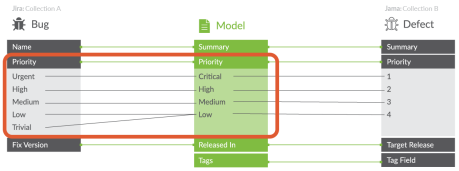
In most single- and multi-select field scenarios, you will configure your transform as '*copy*' on both the collection and on the model side. This means that the model will pass an identical copy of its value to the collection, and vice versa. This should be the default setting.

# Field Value Mapping

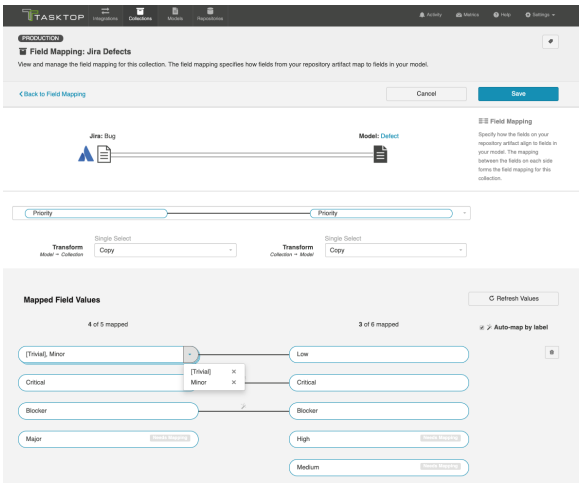
If the **Auto-map by label** box is checked, Tasktop will use its built-in smarts to pre-map some of the field values for you based on their labels. If you'd like, you can click the trash can icon next to each mapping to remove the mapping, and then manually re-map them.

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.

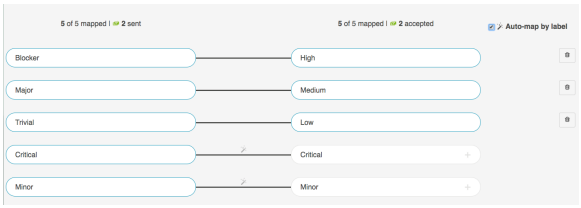



💡 When you map multiple collection values to a single model value, you will find that one value on the collection side is listed in brackets. This indicates which value will be written when the mapped model value is flowed to that field. In the scenario below, if the model passes a 'low' priority value 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.

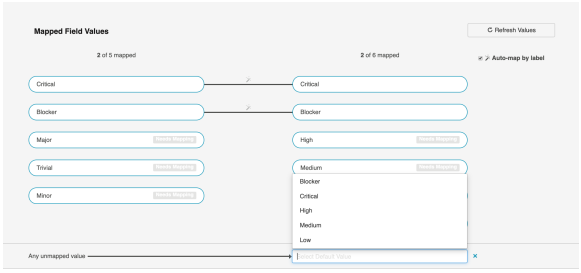



💡 **Note**: If your model allows unmapped values to flow for the field you are configuring, you will see an indication of both the number of values that are explicitly mapped to your model, and the number of values that have been 'accepted' by your model. The values that have been 'accepted' are those 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.



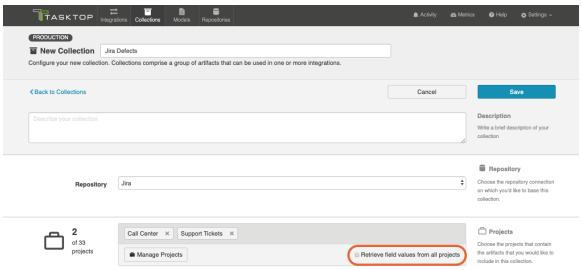

## Default Field Value Mapping

You can set a default field value for any unmapped value by going to the bottom of the **Field Mapping** s creen 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

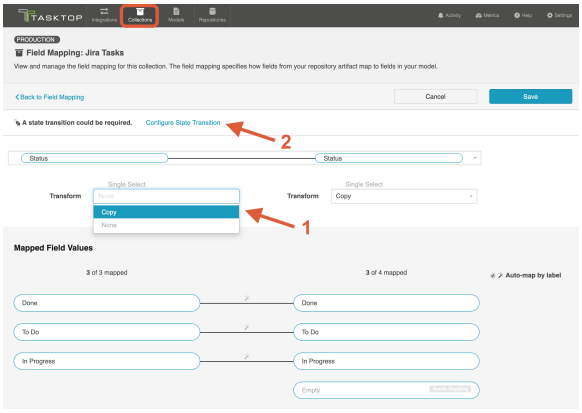
There are a few specific scenarios that will require additional configuration

## 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.

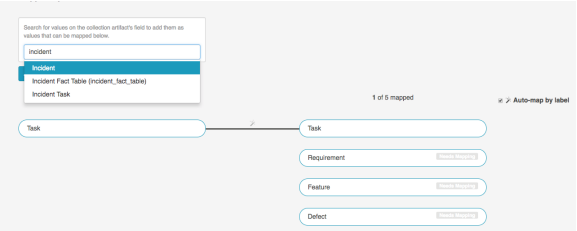
You can learn more about how to configure the state transition on the State Transitions page.

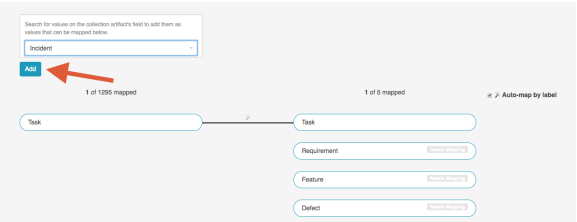


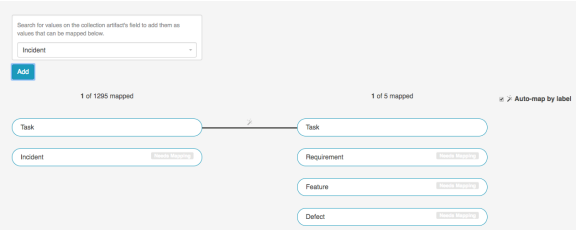

## Single- or Multi-Select Fields with 25+ Possible Values

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.



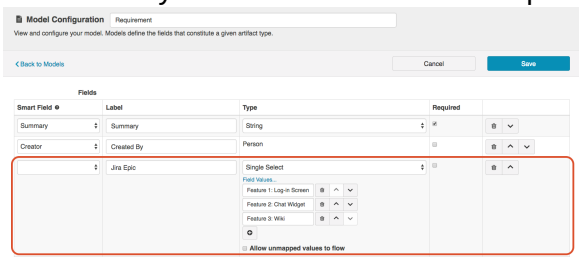



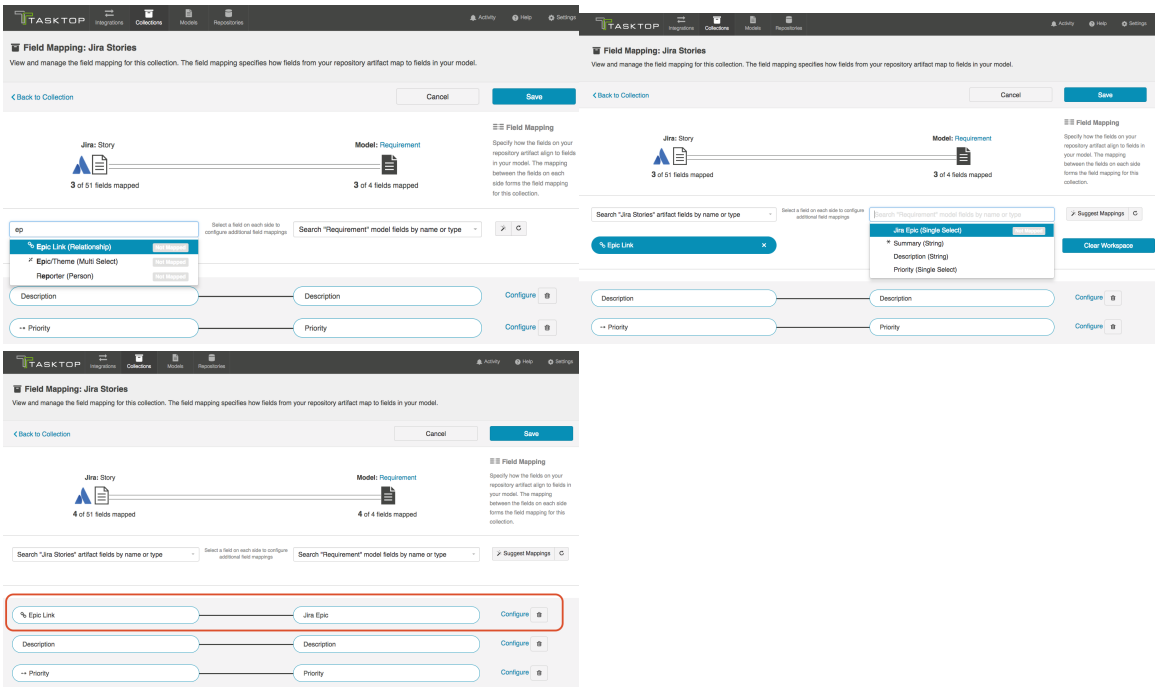

## Relationship to Single-Select Transform

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

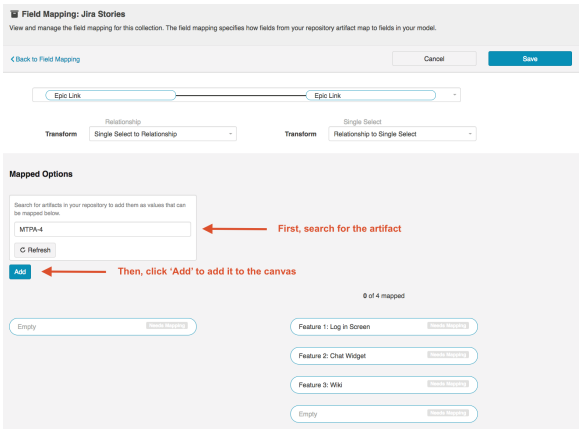



2. In the source collection, click **Map Fields** and create a mapping from the collection's relationship field (Epic-Link in this example) to your model's single-select field.

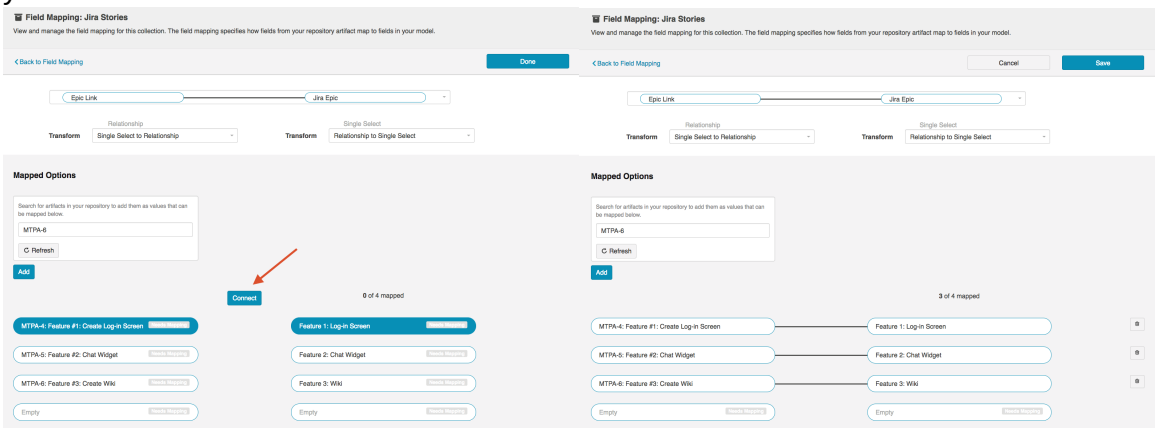



3. Once the fields are mapped, click the **Configure** link on the right side.
4. Here you can search for the related Epics by their **formatted ID**, and click **Add** to add them to your canvas.

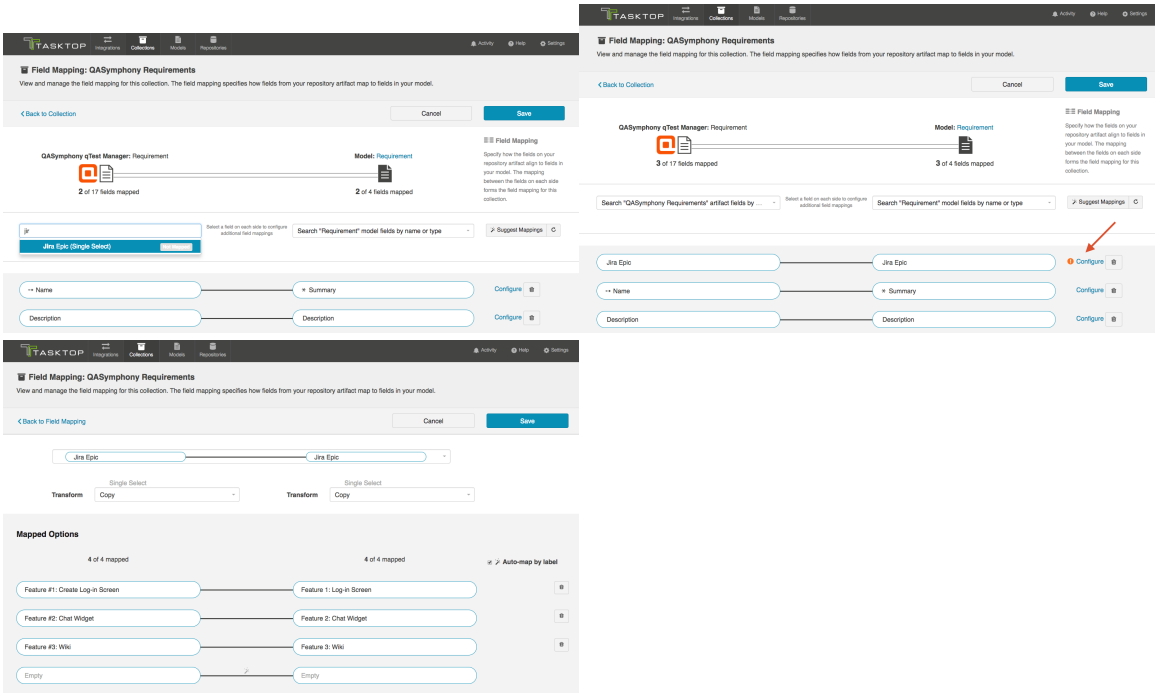
   **Note**: if the artifact you are looking for has recently been created in your repository, click the **Refresh** button to refresh the artifacts that Tasktop is aware of. This will allow Tasktop to find that artifact.



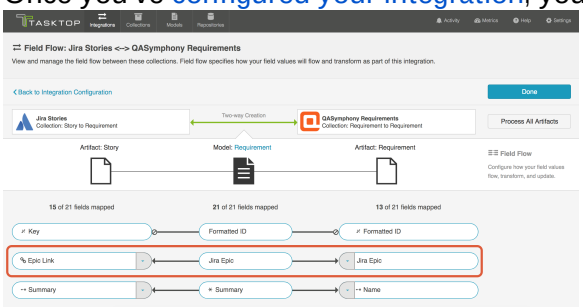
5. Once the related Epics are added to the canvas, map them to the available single-select fields in your model.




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

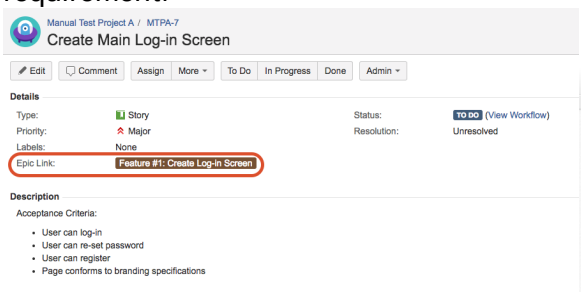


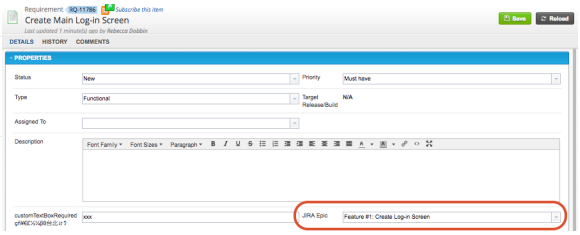

9. Once you've configured your integration, your completed Integration Field Flow will look like this:




10. 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

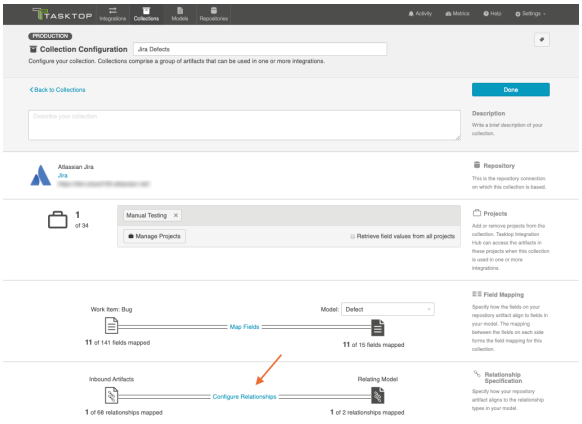
## Introduction

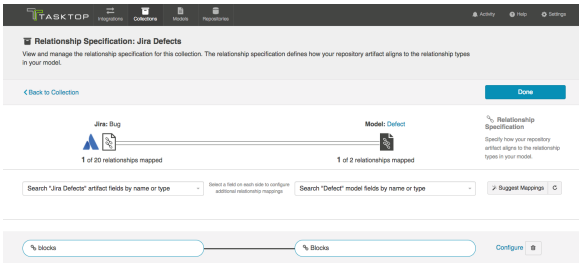
Once you've completed your Field Mapping and Field Configuration, your next step is to configure your Relationship Specification.  The Relationship Specification screen will allow you to specify how **relationship** fields in your repository are mapped to fields in your model.  Relationship fields, such 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.

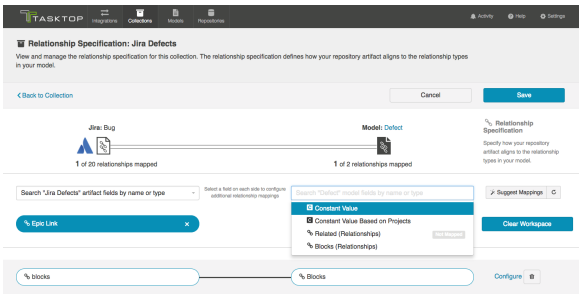
💡 Note that any relationship(s) types you'd like to flow as part of your integration must be mapped to **each** collection involved in the integration.




## 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.

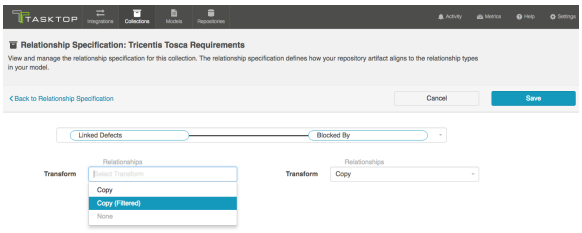



You can learn more about Artifact Relationship Management (ARM) here.

## 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.

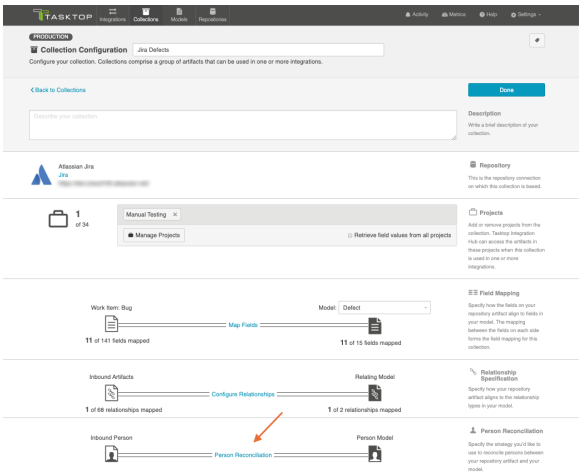
# Person Reconciliation

## 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:

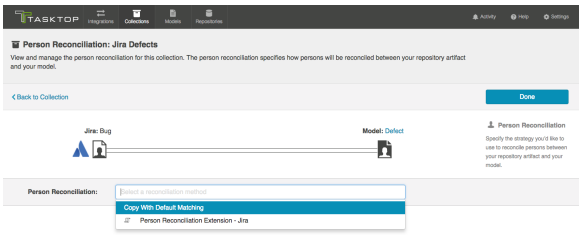
- Username from source to username on target
- Username from source to ID on target
- ID from source to username on target
- ID from source to ID on target
- Email from source to email from target

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.




# 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

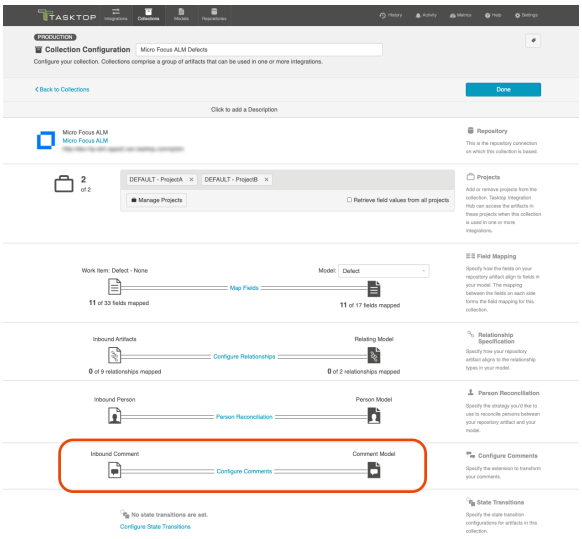
Once you have completed your Person Reconciliation configuration, your next step will be to review your collection's Comment Configuration. On this screen, you will be able to apply an extension to comment handling for your collection.

## Configuring Comments

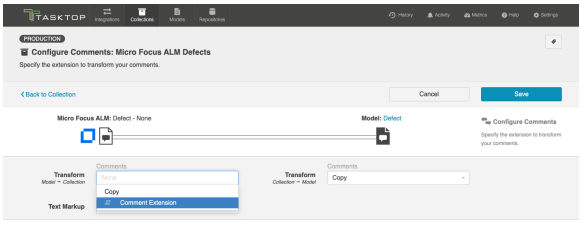
First, create and save your comment extension on the Extensions (Settings) screen. It will be a **custom data transformation** extension.

💡You can learn best practices for configuring a comment extension here.

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

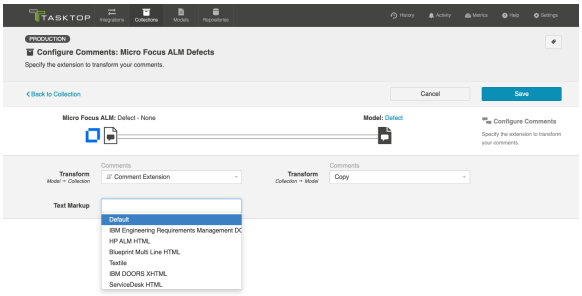



Select the comment extension on the desired side (either impacting data flowing from model  collection or collection  model):

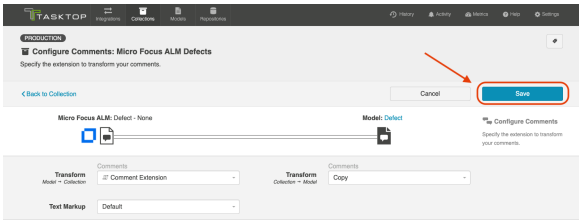



You can also select the text markup language to preserve rich text when flowing comments into and out of your collection.

💡**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.




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

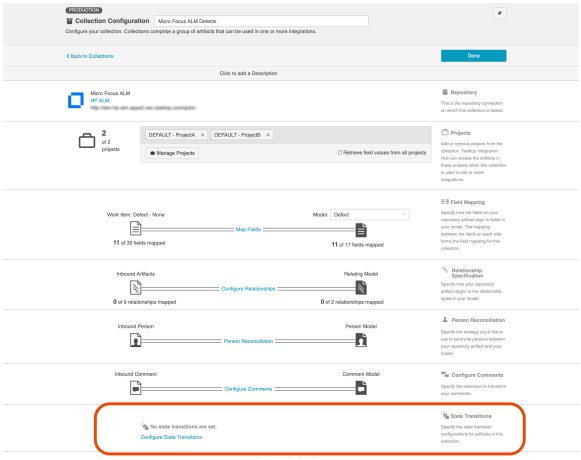
Once you've configured your Person Reconciliation strategy, your next step will be to configure State Transitions, if your repository utilizes state transitions or workflows.

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.

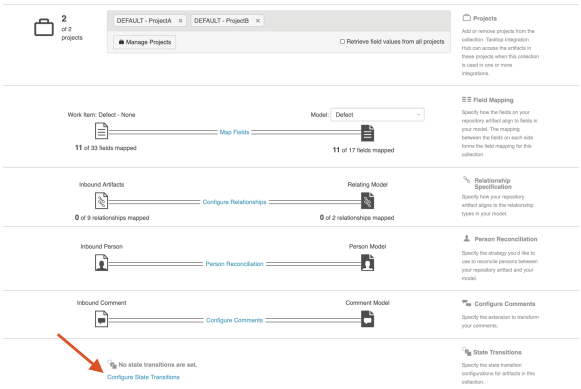
You can also review our Connector Docs to see if state transitions are supported for the repository you are connecting to.

## Configuring State Transitions

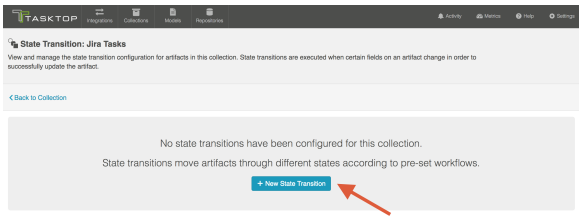
If state transitions are supported for your repository, you will see a **State Transition** sash at the bottom of the **Collection Configuration** screen.




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



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

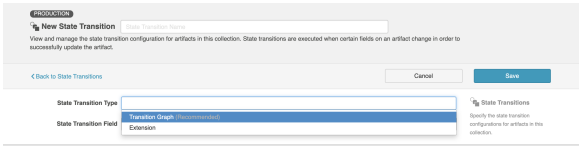



This will lead you to the **New State Transition** screen. Here you can name your transition and choose between two State Transition Types:

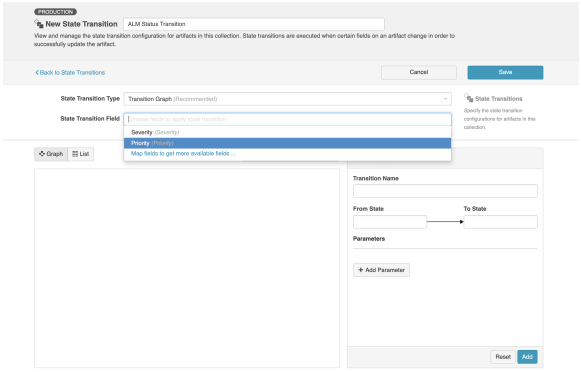
- Transition Graph (Recommended)
- Extension


# Transition Graph

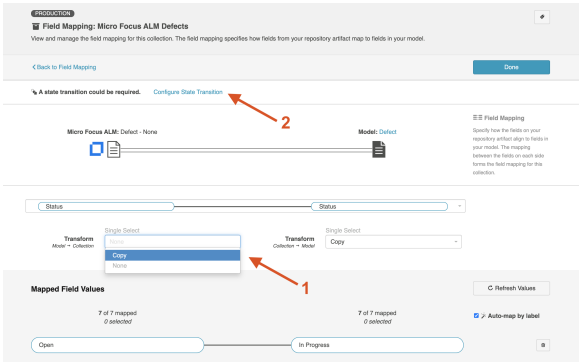
To configure state transitions within Tasktop's UI, select **Transition Graph** as your State Transition Type.




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

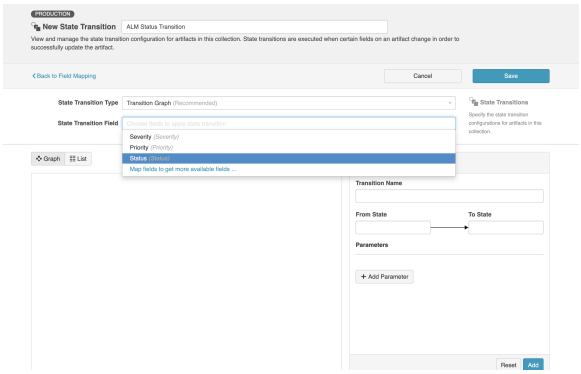



If you don't see the field you'd like to use, make sure that the field is **mapped** and that **its transform is set to copy on the repository side**. Once you set the transform to **Copy**, you will see a **Configure State Transition** link. Click that link to return to the State Transition screen.

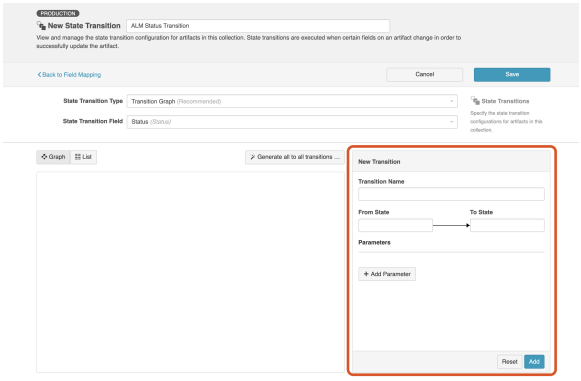



Now you can select the field on the New State Transition screen:

💡 **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.



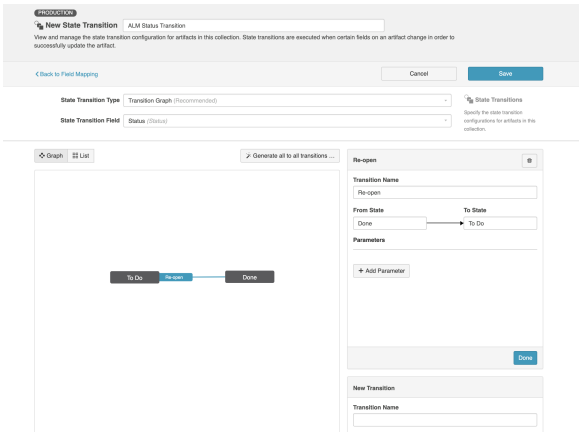

Now that you've selected your field, you'll see the Transition Configuration panel.




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.

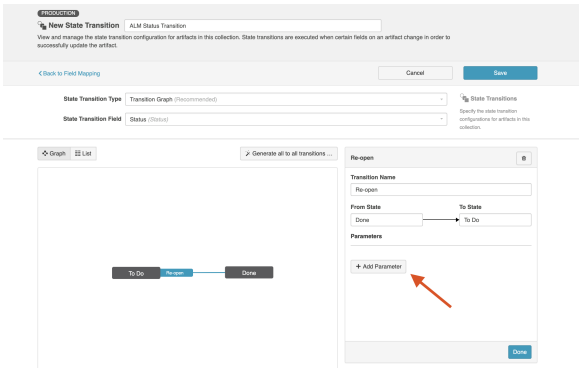
Here is an example of a transition that has been configured. Note that when you view a transition (by clicking on it in the graph), you'll see its configuration on the right so that you can make any needed modifications. You'll also see a **New Transition** pane immediately underneath, so that you can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

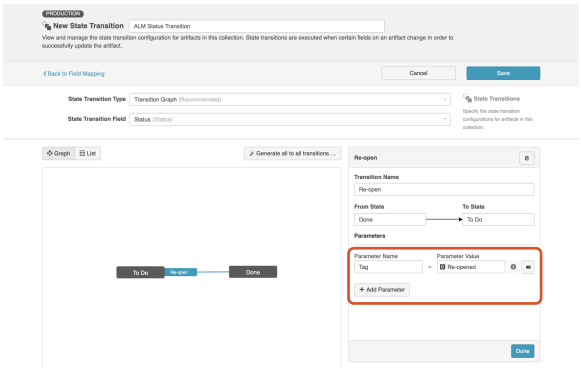
## Adding Parameters

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

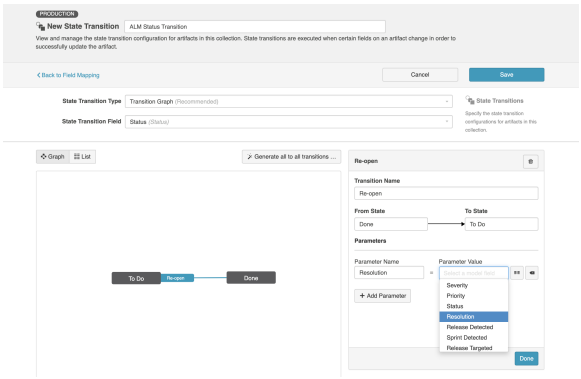



The Parameter name must match the field name within the **repository** exactly. You can either set a constant value for your parameter, or configure the transition to flow a value based on your field mappings.

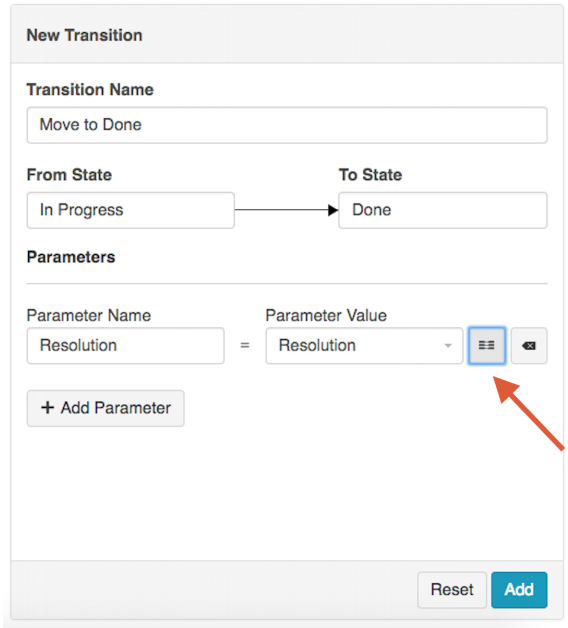
In the image below, we've set a constant value, which will tell Tasktop to add a **Re-opened** tag to the artifact when it moves through the **Re-open** transition:



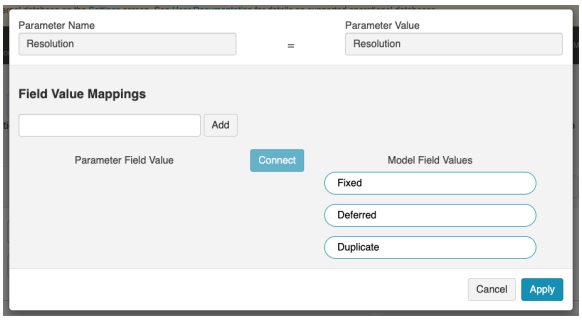
You can also set a Parameter that is set based on a field in the **model**.



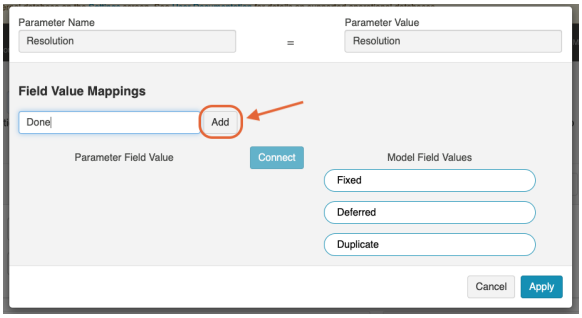
To map the field, click the **map** icon.

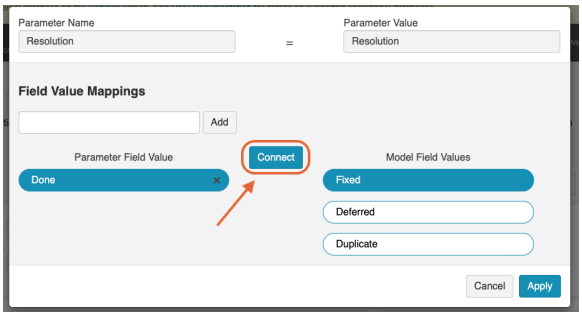



This will bring you to the Parameter Field Value Mappings pop-up.

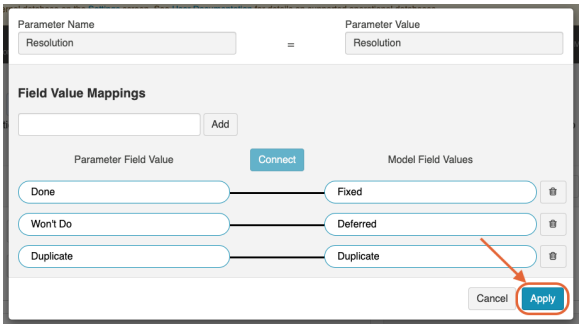



Here you can manually enter the parameter field values on the left that exist within your repository, and map them to the model fields on the right. The field values entered must match the field values that exist in the repository **exactly** (they are case- and space- sensitive).




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

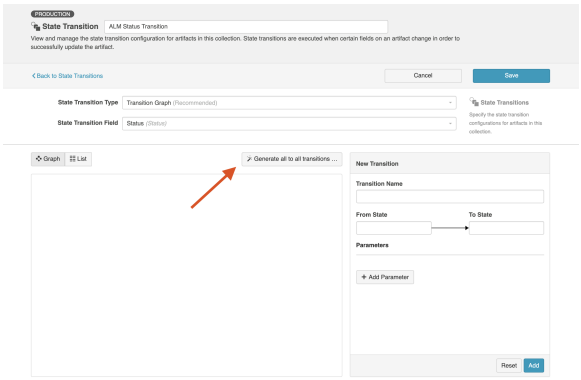



## Generating All to All Transitions

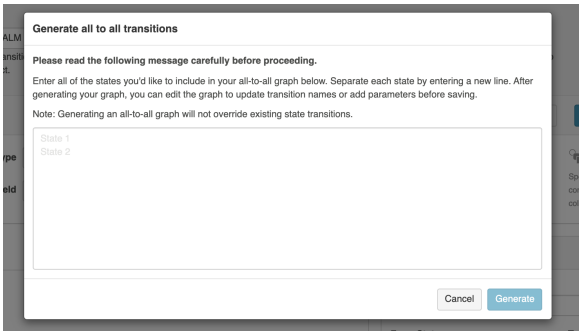
Sometimes workflows require all states to transition to all states and depending on how many states the workflow has, this can result in much manual effort (e.g., for 15 states, you need to create 210 transitions). Rather than manually creating these transitions, Tasktop does most of the work for you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

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

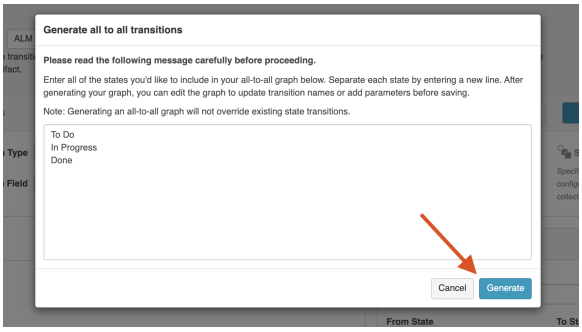
💡 **Note**: Generating an all to all transition graph will not override any existing transitions.



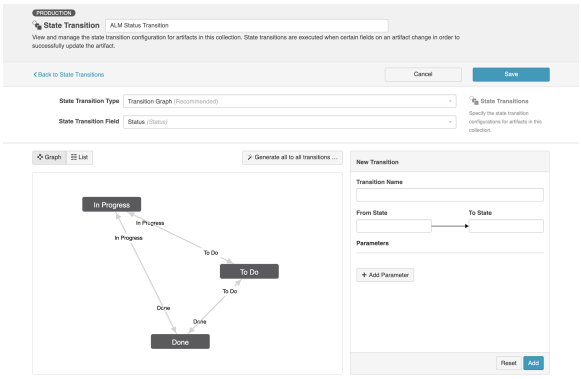

A pop up will appear where you can enter all of the states you'd like to include in your graph.



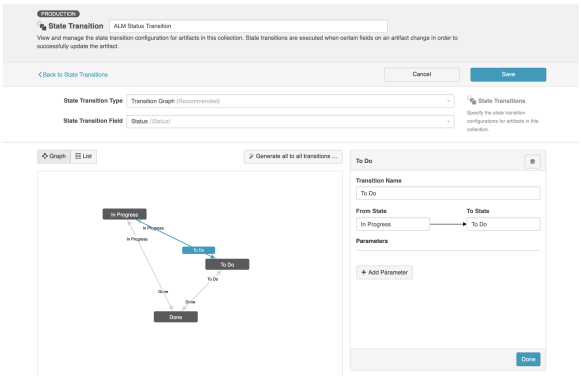

After you've entered all of the states, click **Generate**.




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

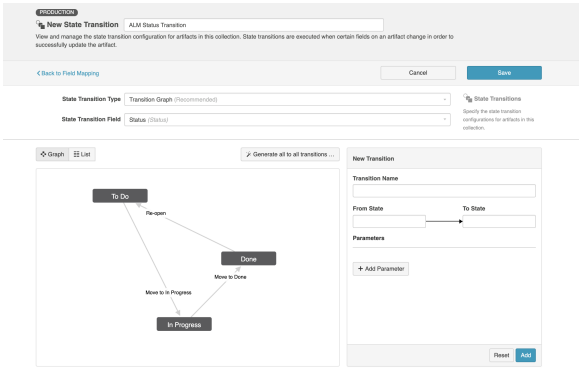



If you'd like to edit a transition, click the transition name and you can edit in the transition configuration panel.



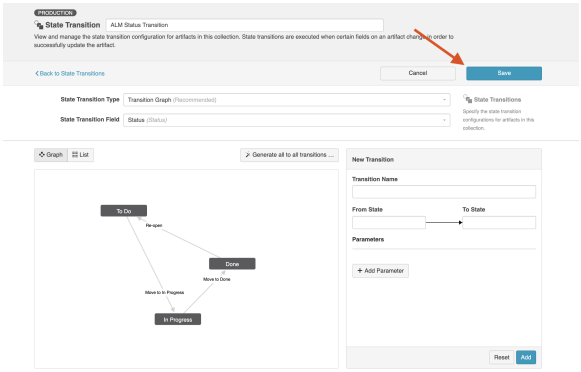

## Saving and Viewing

Here's an example of a completed Transition Graph:

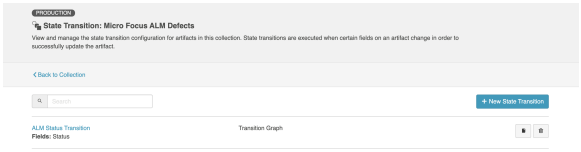



Make sure that your completed graph matches the state transition configuration in your repository **exac tly**. If it does not match, you'll see errors when running the integration.

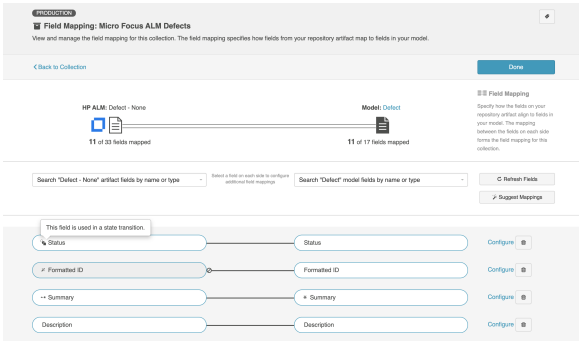
Once confirmed, click **Save** and **Done**.




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



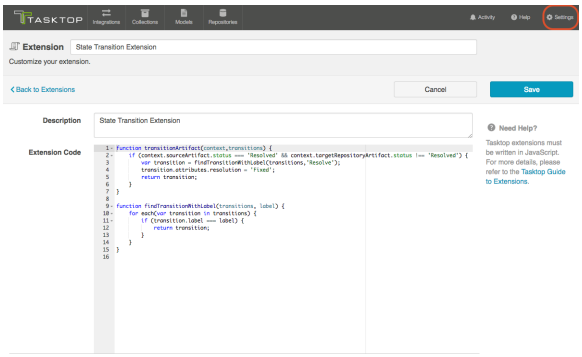

You'll also notice a state transition icon on the collection pill on the Field Mapping screen, denoting that a transition graph is in use.



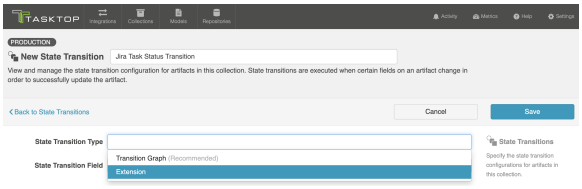

# Extensions

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

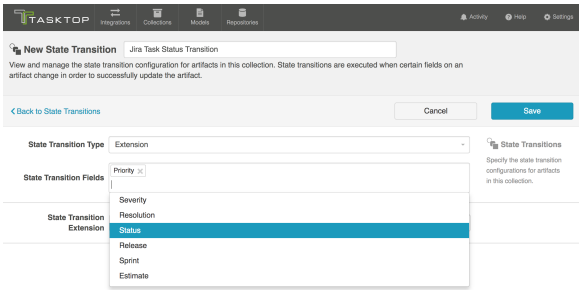
If you choose to configure state transitions via an extension, rather than utilizing the transition graph, your first step will be to create and save the extension itself from the Extensions (Settings) screen. If you need help creating the extension, you can find more information in the Extensions (Settings)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



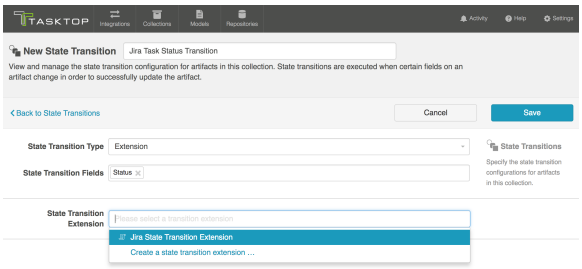

Once the extension is configured, you can select **Extension** as the State Transition Type on the New State Transition screen.



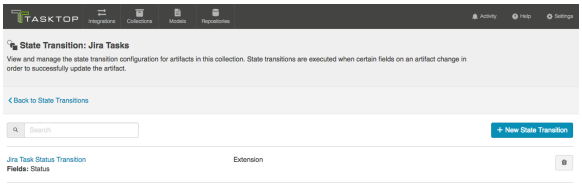

Next, select the **model** field(s) that you'd like to apply the extension to.



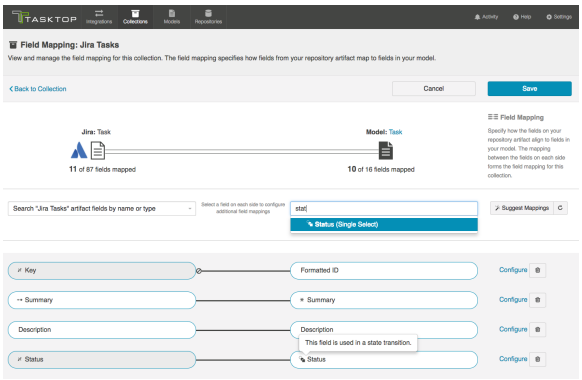

You can then select the extension you'd like to use.



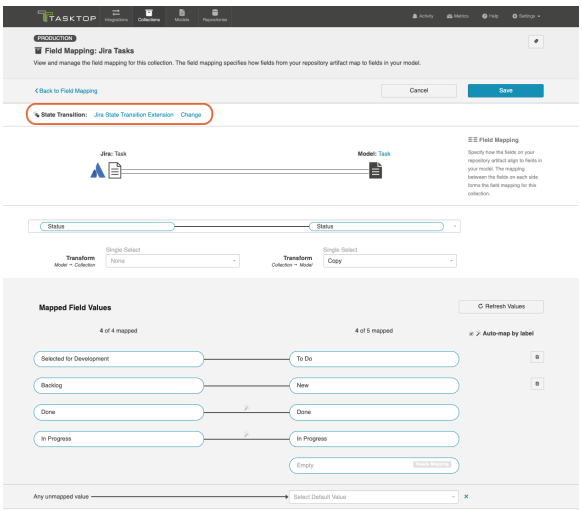

Click **Save** and then **Done**. You'll now see the State Transition Extension listed on the State Transition screen.



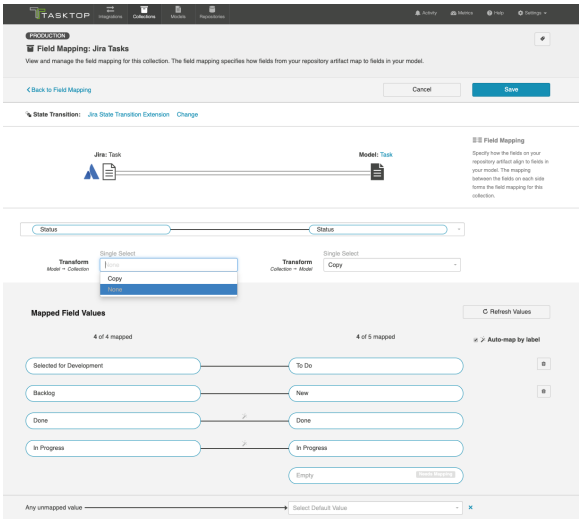

And you'll notice the state transition icon on the model pill and the model drop-down on the Field Flow screen.




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.

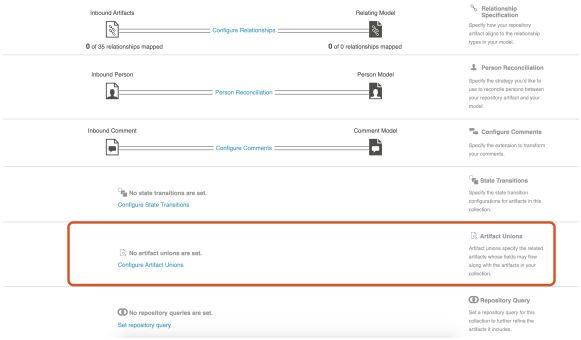
By creating an artifact union, they are able to extract this field from the 'Epic' artifact so that it will flow alongside their Jira Stories collection. After they create the artifact union, it can be mapped to the model where it will flow into their Jama stories collection.

Learn more on how to create and use artifact unions in the sections below.

## Creating Artifact Unions

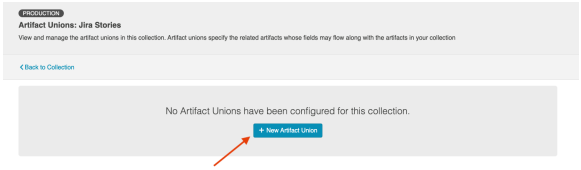
To access your artifact unions, navigate to the **Collection Configuration** screen and click **Configure Artifact Unions**.

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



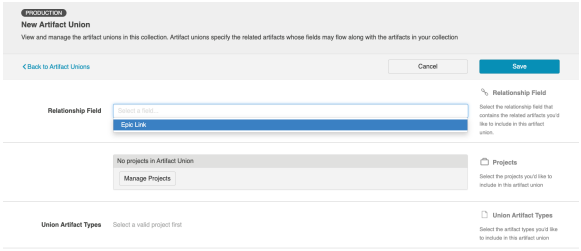

To add a new artifact union, click **+ New Artifact Union**

⚠️ 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.

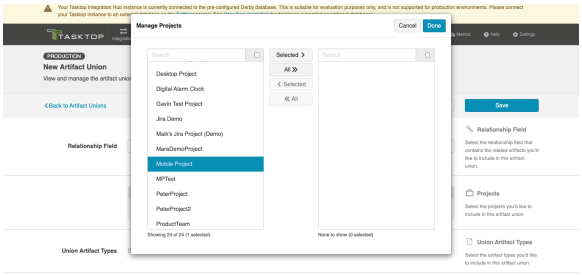


Select the relationship field that links the collection to the target artifact type you'd like to include in your artifact union.

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

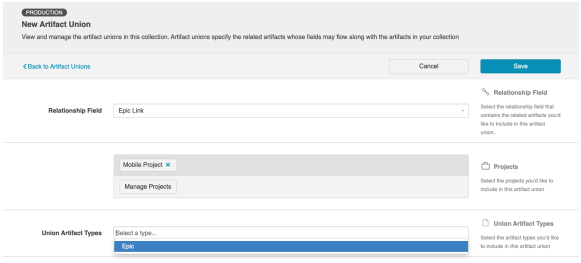



Select the project(s) you'd like to include.

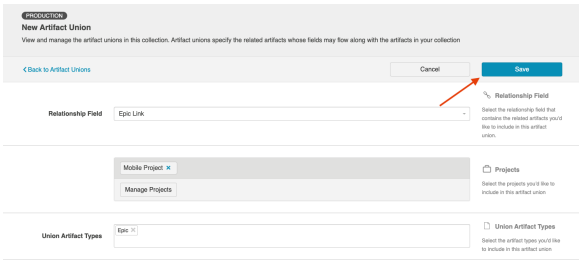



Once you have selected the relationship field and project(s), you can choose the artifact type(s) you'd like to include in your artifact union.

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




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

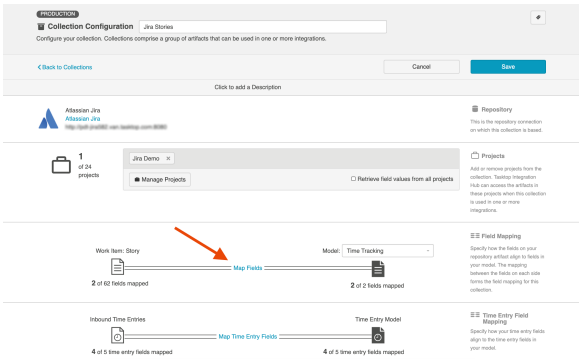


# Utilize your Artifact Unions

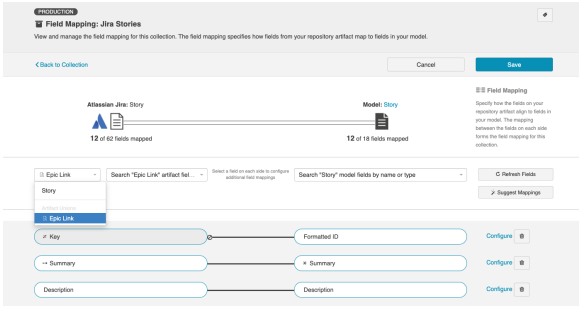
Now that you've created your artifact union, you can now use your artifact union to flow the desired field to the other side of the integration.

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

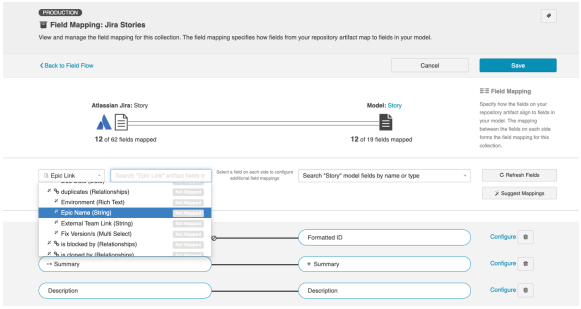
To do this, navigate to the Collection Configuration screen and click **Map Fields**.



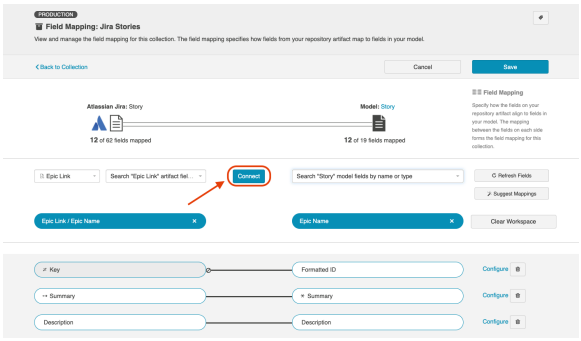

Next, select the configured artifact union from the dropdown menu.



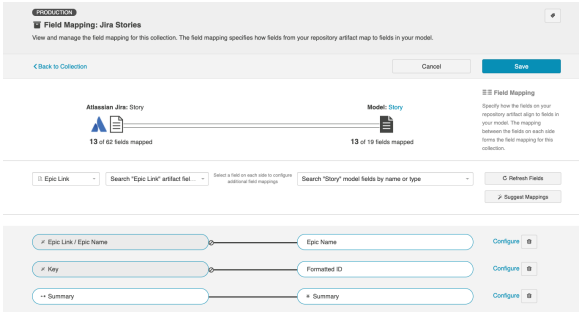

After you've chosen your artifact union, select the field you'd like to extract from the related artifact.




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**.




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

# ALM/Octane Test Management

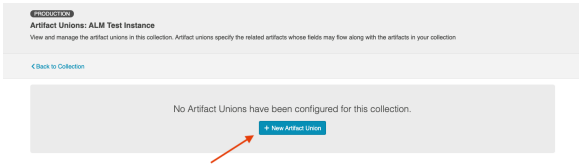
Tasktop Integration Hub offers integration solutions to flow test artifacts such as test results, test steps, and their associated tests, test runs, test instances, and folder structures in Micro Focus ALM and Micro Focus ALM Octane. However, due to complex containment structures and multiple hierarchy 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.

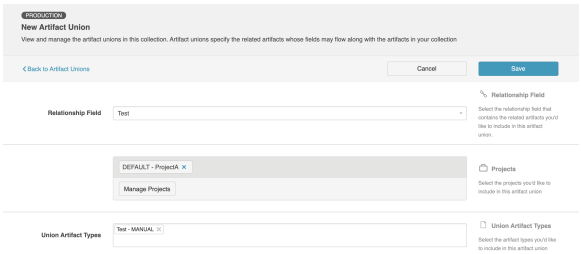
In the example below, the user has an 'ALM Test Instance to Octane Tests' integration and they'd like to flow 'Test Steps' from ALM into their Octane Tests collection. However, ALM Test Steps only exist on the ALM 'Test' artifact. Because the ALM Test Instance has a shared relationship field with ALM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

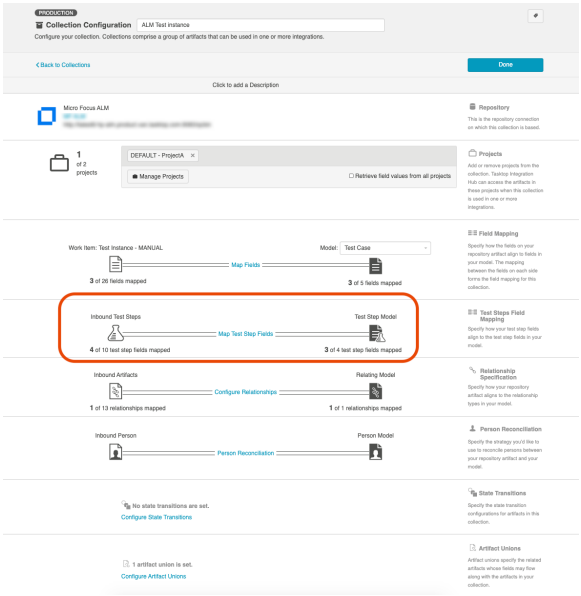
First, the user navigates to their ALM Test Instance collection and creates a new artifact union.



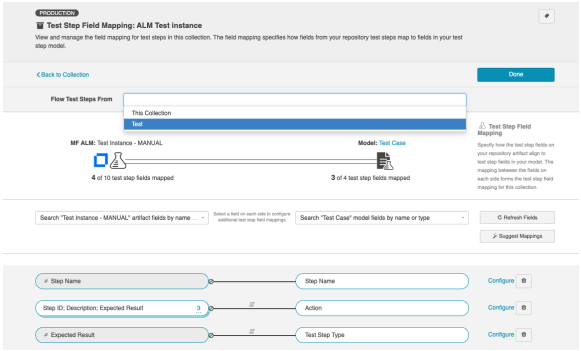

Then, the user selects the **Test** relationship field, their desired project, and artifact type.




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.



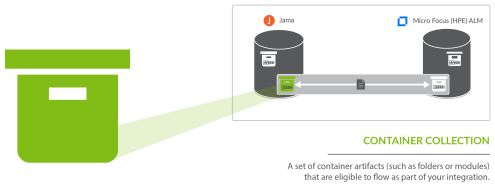
After saving, the user can flow the ALM test steps into their Octane tests collection. And that's it!

## 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

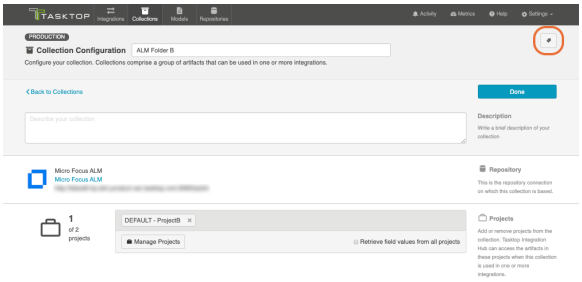
The steps to configure a Container Collection are very similar to the steps to configure a Work Item Collection (Repository). Please refer to that page for in depth instructions.

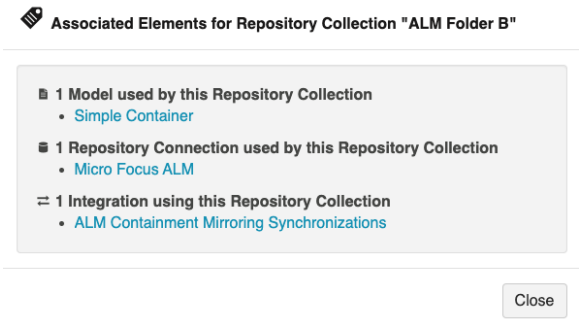
You will, however, notice a few key differences:

- 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.

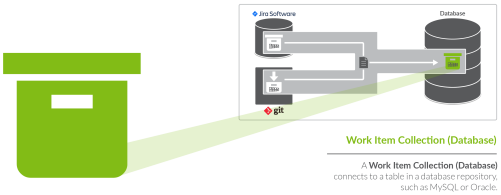



Associated Elements for Repository Collection "ALM Folder B"
1 Model used by this Repository Collection
Simple Container
1 Repository Connection used by this Repository Collection
Micro Focus ALM
1 Integration using this Repository Collection
ALM Containment Mirroring Synchronizations
Close

# 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

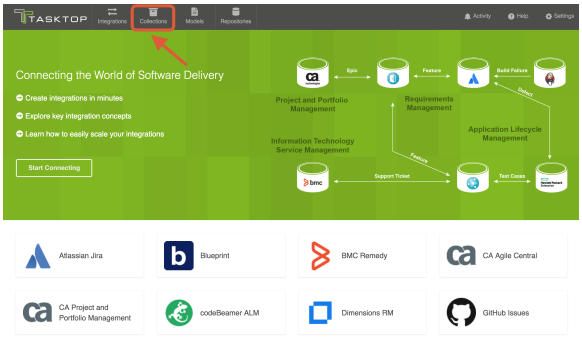
Check out the video below to learn how to create a new collection for your database repository:

💡 ***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.*

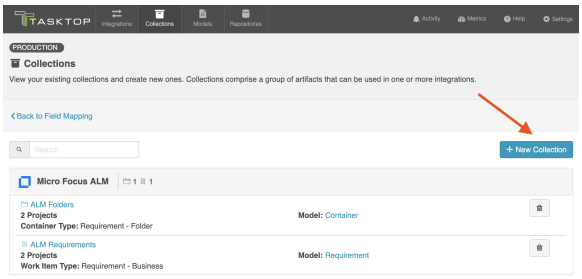
## Creating a Database Collection

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

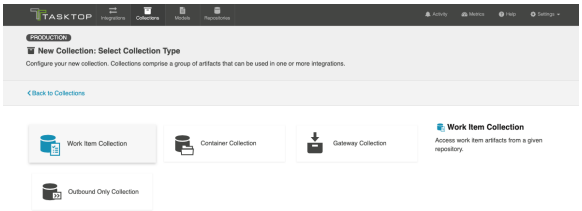
Select **Collections** at the top of the screen.



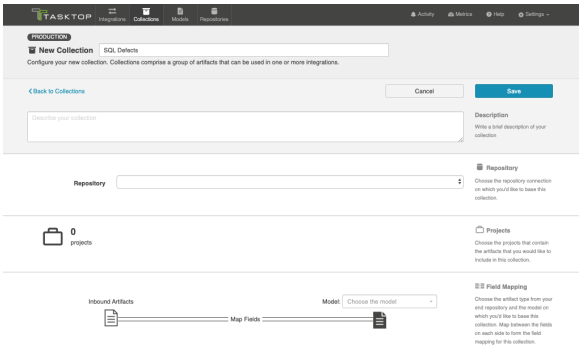

Click **New Collection**.



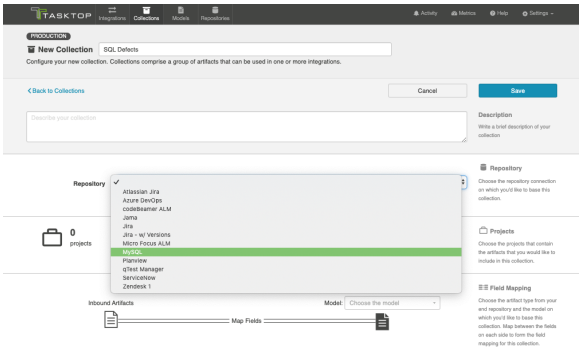

Select **Work Item Collection** as the collection type.



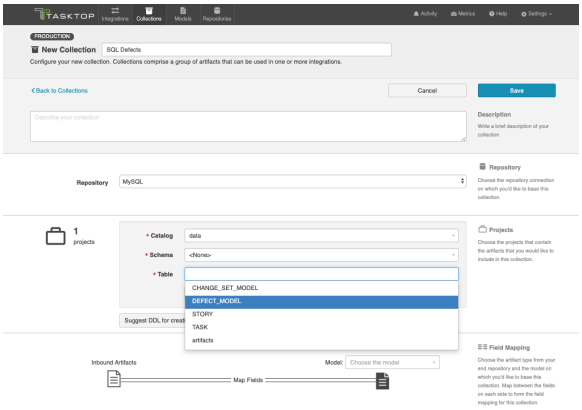

Enter the name for your collection.



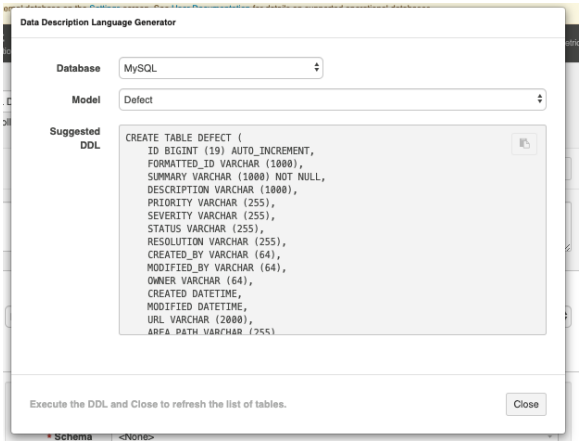

Select the Connection on which you'd like to base this collection. In our example, we are selecting MySQL, which is the **Tasktop SQL** repository connection we have configured.



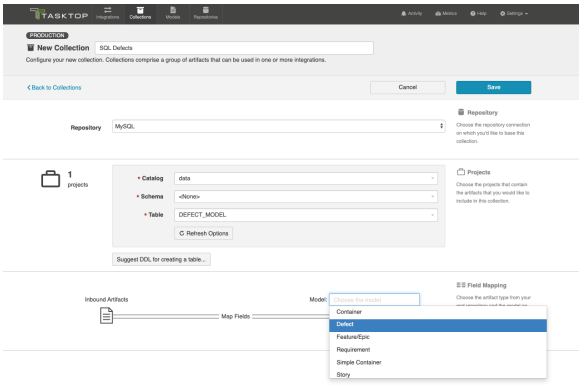

Select the database table that will receive artifacts that flow to this collection.

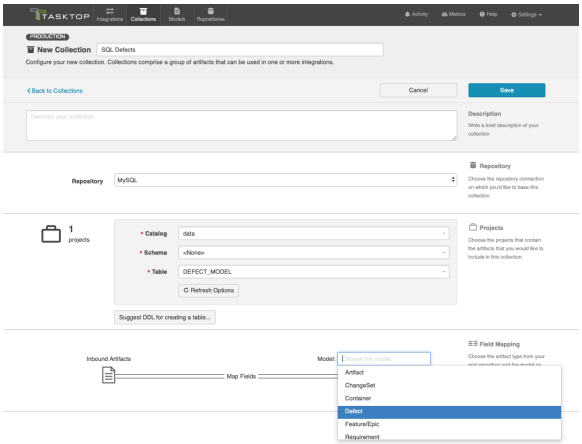



💡 **Note**: if your table is not listed, you can use the **Suggest DDL** tool to generate a SQL command that can help you create a table that aligns with the model on which you'd like to base this collection.




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

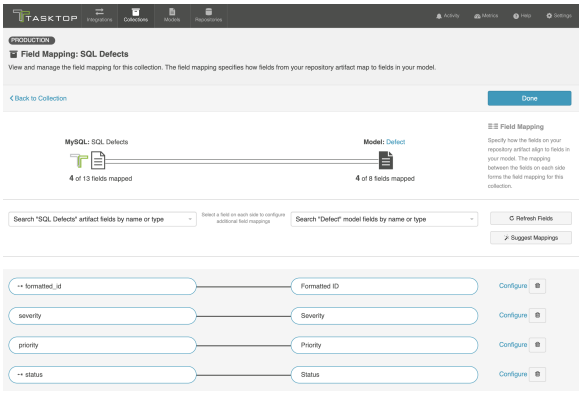



# Map Fields

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

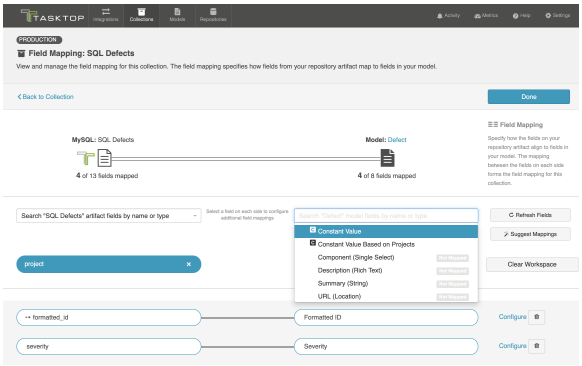
in the model that can provide a value, you can set a constant value into your end-database column /field. The value you configure will then always get written out.

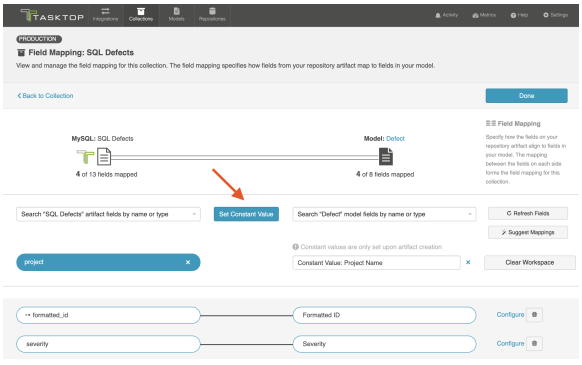
To set a constant value for a field, select the **Constant Value** option from the drop down menu on the model side. This will tell the integration to *always* flow that value to the database collection. Enter the value, and then click the **Set Constant Value** box.

💡 **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.

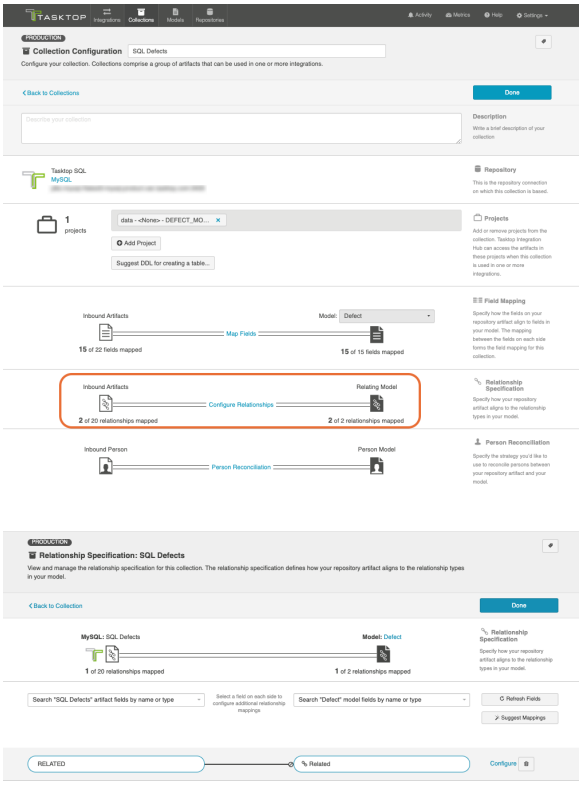



## Configure Relationships

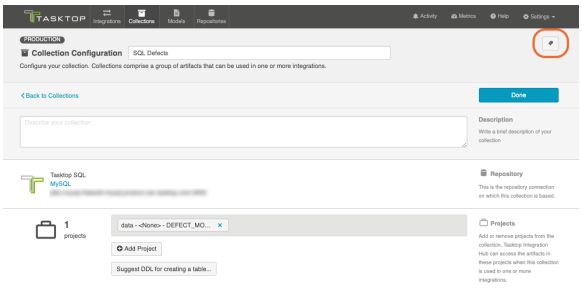
If you have any relationship(s) fields in your model, you can map those on the **Configure Relationship Types** screen of a given collection.

💡 **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.

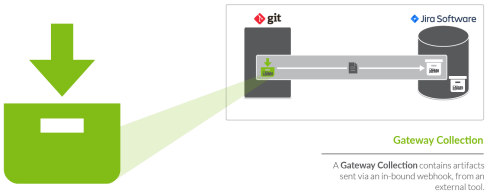



Associated Elements for Repository Collection "SQL Defects"
1 Model used by this Repository Collection
Defect
1 Repository Connection used by this Repository Collection
MySQL
Close

# 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.

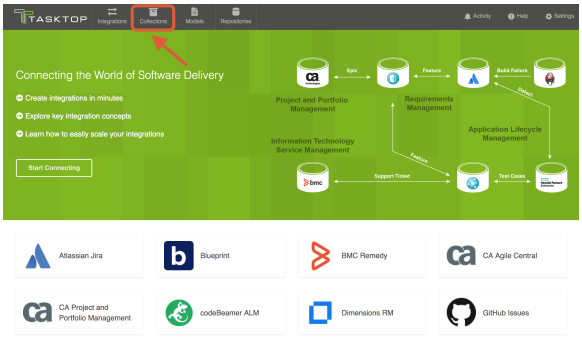
## Video Tutorial

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

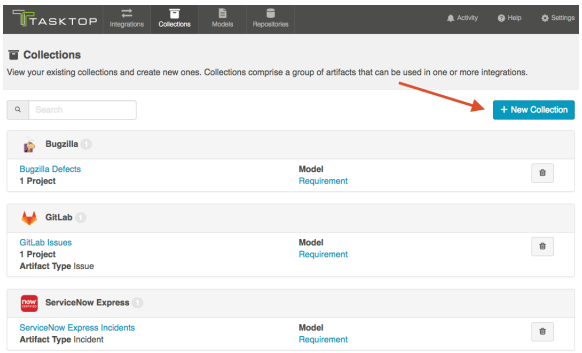
## Creating a Gateway Collection

To create a gateway collection, follow the steps below.

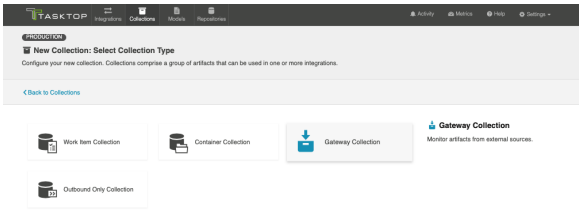
Select **Collections** at the top of the screen:



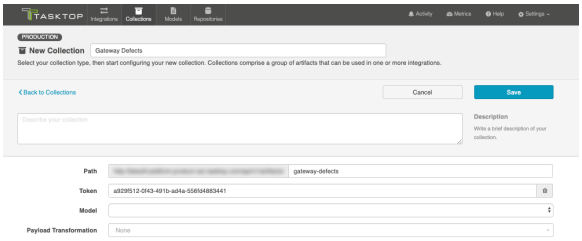

Click **New Collection**.




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

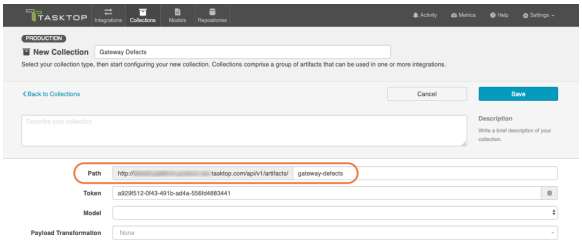



Enter a name for your collection.



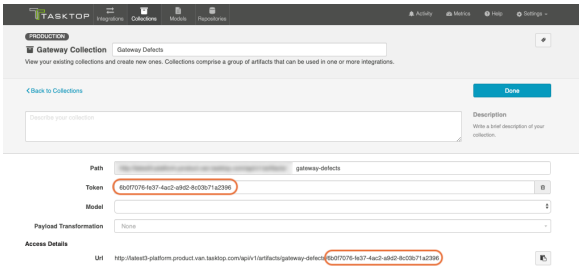

Next, specify the **path** for your collection. These characters will form the REST endpoint to which you can send artifacts to Tasktop via this gateway collection.

💡 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.

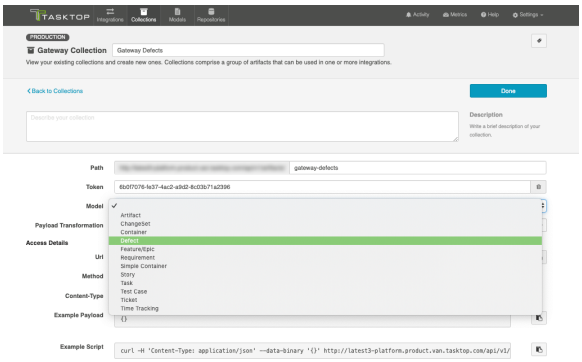



To **secure your gateway collection**, Tasktop automatically appends a token (a universally unique identifier) to the path of a gateway collection. This token will be incorporated into your gateway URL and will help ensure that only users that know the full path with its token can access your gateway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

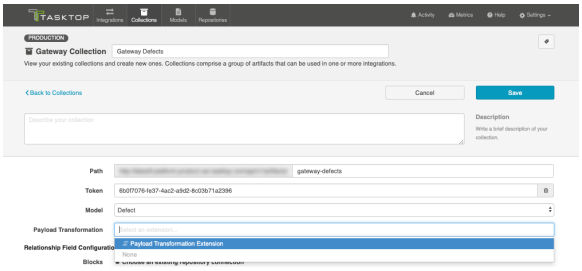
If using Tasktop On-prem, you can remove the token by clicking the trash can icon to the right, and refresh it by hitting the magic wand icon that appears in its place (for security, tokens cannot be removed on Tasktop Cloud). Once refreshed, click **Save**, and the URL will be updated.



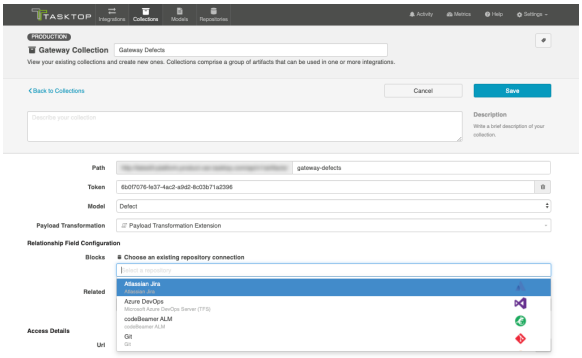

Select the model on which you'd like to base the collection.



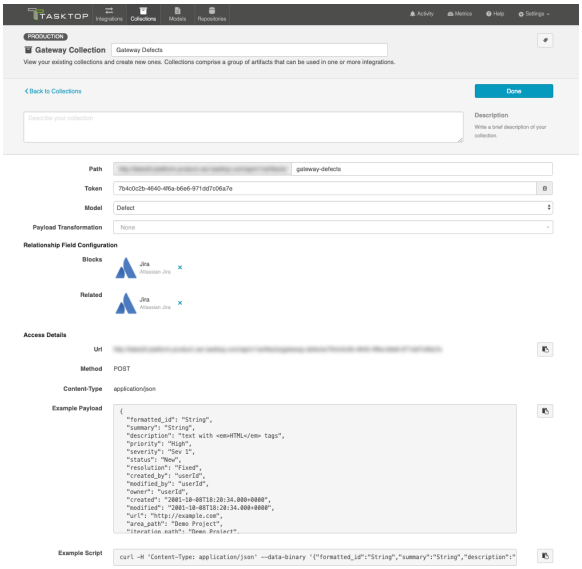

If you have configured a payload transformation extension for your gateway collection on the Extensions (Settings) screen, you can select it here.




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.



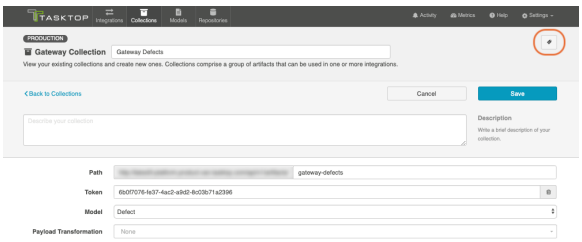

Once you've saved your collection, you will be able to observe the access details given for this gateway collection.

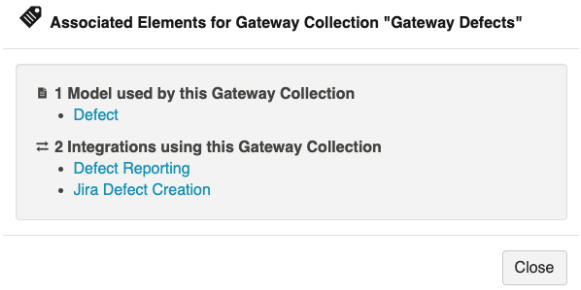



The example payload can be used to construct the JSON payload that will be sent to Tasktop from your external tool.

# 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.

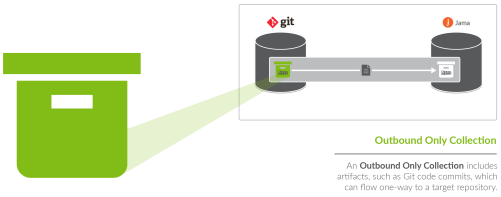



Associated Elements for Gateway Collection "Gateway Defects"
1 Model used by this Gateway Collection
Defect
2 Integrations using this Gateway Collection
Defect Reporting
Jira Defect Creation
Close

# 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.

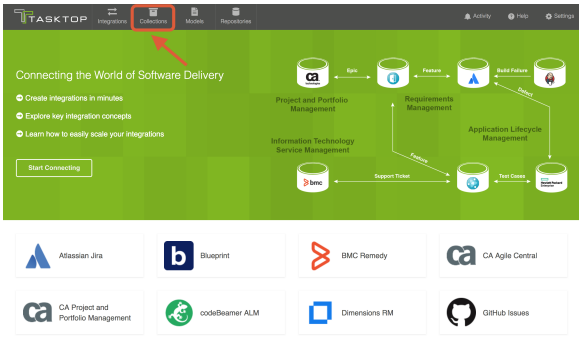
You can learn more about collections in the Key Concepts.

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

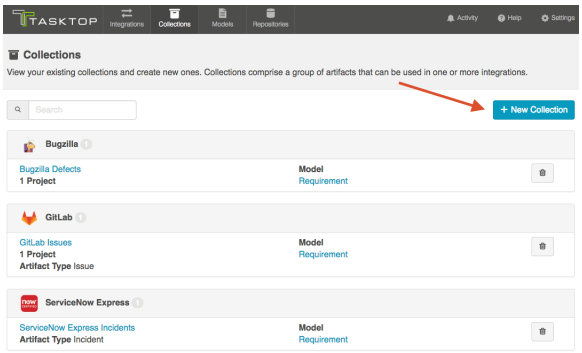
## Creating an Outbound Only Collection

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

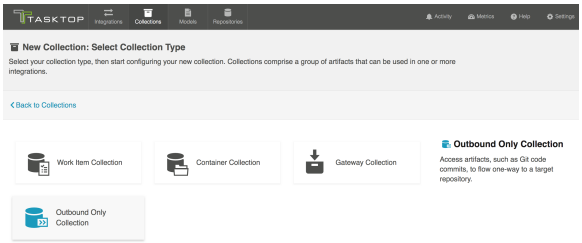
Select **Collections** at the top of the screen.



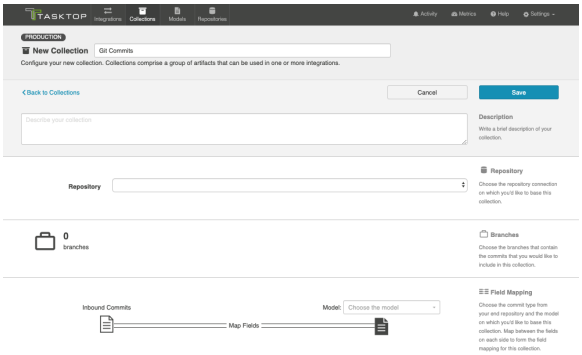

Click **New Collection**.



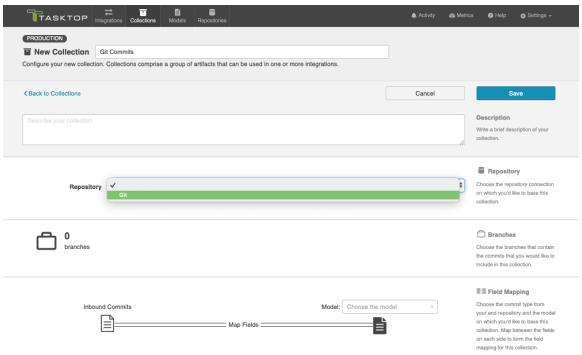

Select **Outbound Only Collection** as the collection type.



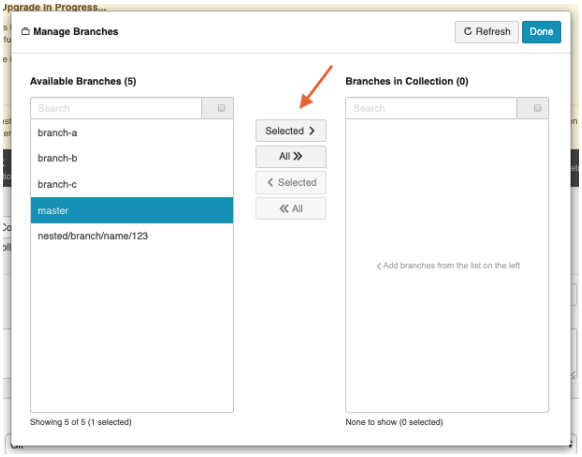

Enter a name for your collection.



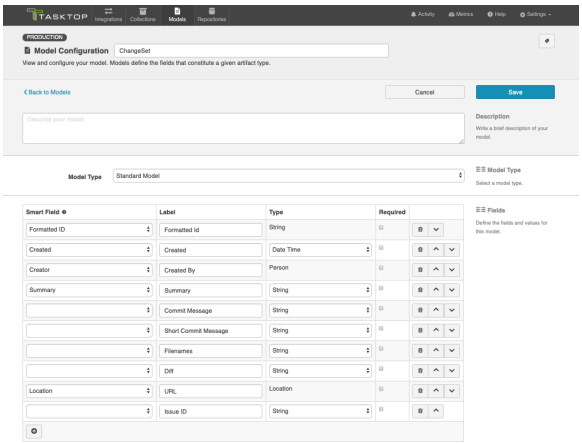

Select the repository that you would like to connect. The Outbound Only collection type can only connect to the Git repository. The collection will include artifacts from the repository you have selected.

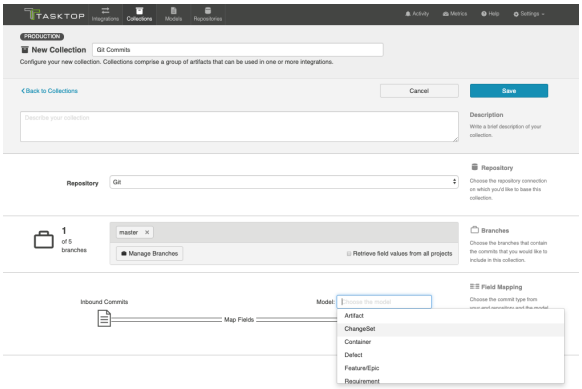



Add branches to your collection by selecting **Manage Branches**. These are the branches from which Tasktop will be flow code commits, changesets, or other artifacts.

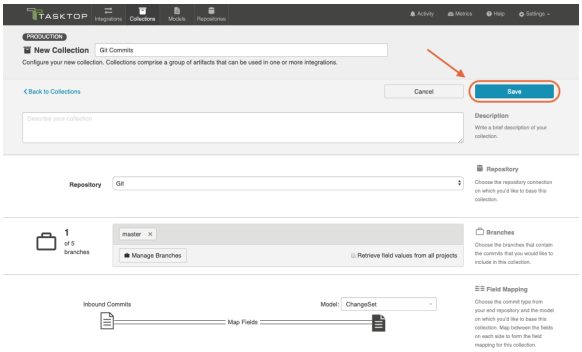



Select the model you'd like to use for this collection. If available, choose the ChangeSet model. We recommend ensuring that your model contains the following fields:

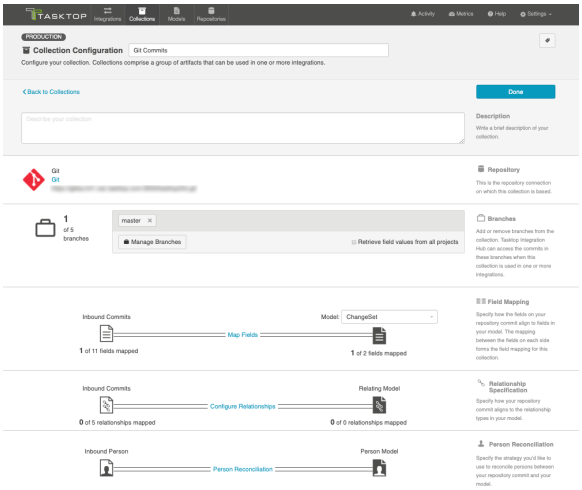



Click **Save**.




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

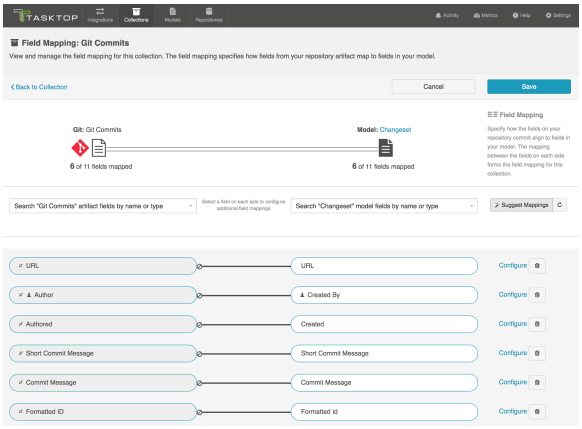



# Map Fields

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

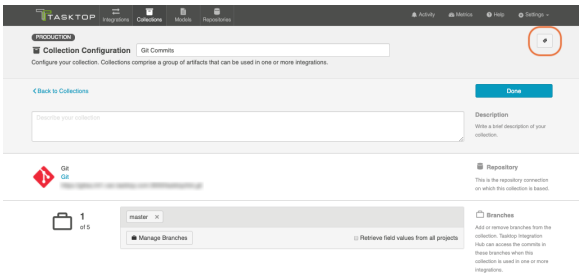
If you have enabled repository queries for the repository that you have connected to, you will also see a **Repository Query** sash at the bottom of the screen.

⚠️ 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.




## 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.

|  **Work Item Synchronization** |  **Container + Work Item Synchronization** |  **Create via Gateway** |  **Modify via Gateway** |  **Enterprise Data Stream** |
|---|---|---|---|---|
| *The Work Item Synchronization template is available in all Editions.* | *The Container + Work Item Synchronization template is available in all Editions.* | *The Create via Gateway template is only available in Editions that contain the Gateway add-on. See Tasktop Editions table to determine if your edition contains this functionality.* | *The Modify via Gateway template is only available in Editions that contain the Gateway add-on. See Tasktop Editions table to determine if your edition contains this functionality.* | *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.* |
| This integration connects teams | This integration connects teams working in | This integration creates traceability between artifacts across the software development | This integration creates traceability between artifacts across the software development lifecycle. Already existing | This integration simplifies enterprise reporting by unlocking software lifecycle data from its application tool silos and |

| | | | | |
|---|---|---|---|---|
| working in different tools as they fulfill their roles in the software development lifecycle.<br><br>As part of this integration, work items will flow between separate repository collections. | different tools as they fulfill their roles in the software development lifecycle.<br><br>As part of this integration, work items will flow between separate repository collections.<br><br>Additionally, the containers in which your work items reside will be mirrored across the collections according to your specification. | lifecycle. New artifacts will be created in a repository collection when artifacts are sent to Tasktop via a Gateway collection, through an inbound webhook. | 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. | 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. |
| Learn More | Learn More | Learn More | Learn More | Learn More |

# Prebuilt Integration Patterns

Base your integration on one of these patterns to address typical business needs.

|  <br>**Code Traceability: Create and Relate a Changeset** |  <br>**Code Traceability: Update an Existing Work Item** |  <br>**Test Synchronization** |
|---|---|---|
| *This integration template is only available in editions that have access to the Git repository.* | *This integration template is only available in editions that have access to the Git repository.* | *See Tasktop Editions table to determine if your edition contains this functionality.* |
| 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.<br><br>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. | 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).<br><br>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. | This set of integrations flows Test Result information on a Test Run in ALM or Tosca.<br><br>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. |
| Learn More | Learn More | Learn More |

# Work Item Synchronization

## What is an Integration?


An *integration* is quite simply **the flow of information between two or more collections**.

A w*ork 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.

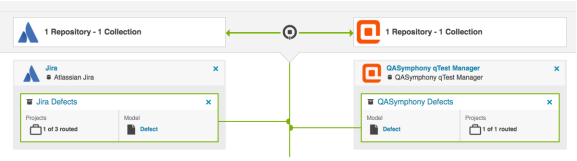
⚠️ This video assumes that you have already configured your repositories, models, and collections as outlined in the Quick Start Guide.

## 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.



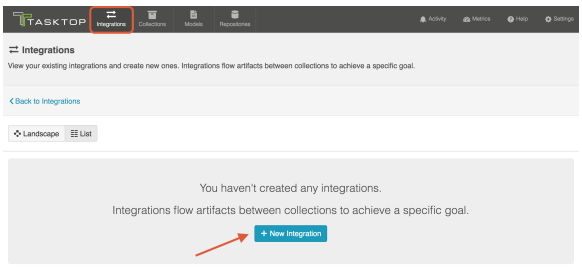

# Template Affordances

The Work Item Synchronization template allows you to flow artifacts between two work item (repository) collections.


# Configuring a Work Item Synchronization Integration

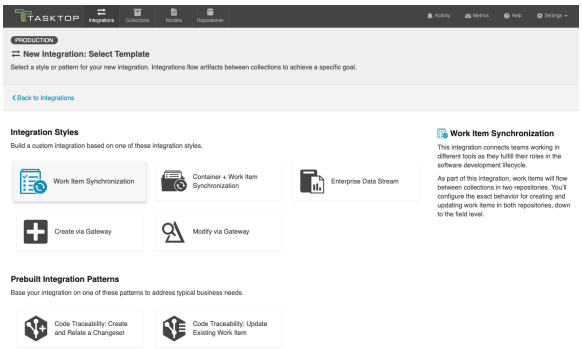
Now that you have all of your base components (i.e., repositories, models, and collections) set up, you can configure an integration to synchronize the artifacts in your collections.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

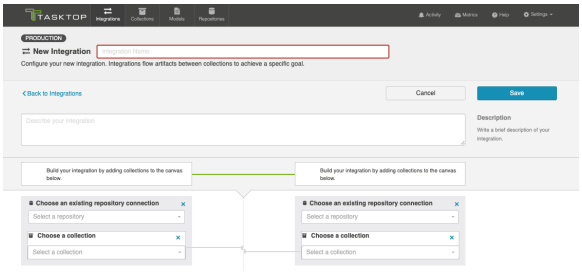



Select the **Work Item Synchronization** template.

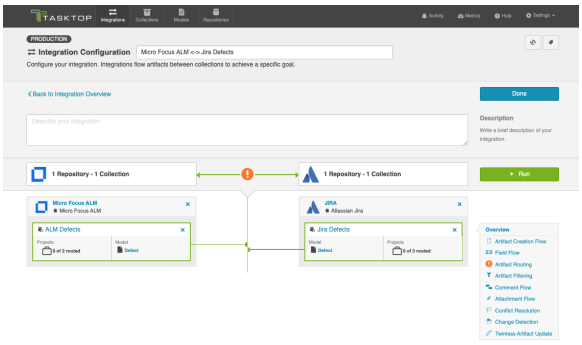
Depending on the edition of Tasktop you are utilizing, you may not have all options available.



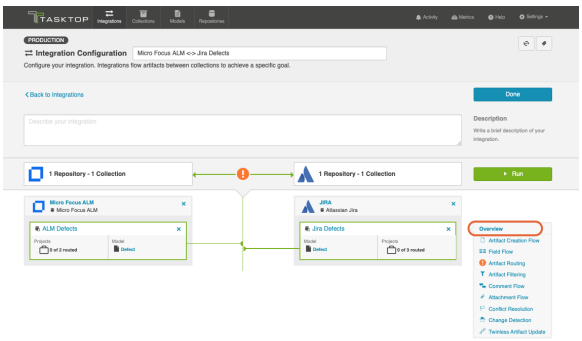

This will bring you to the **New Integration** screen.

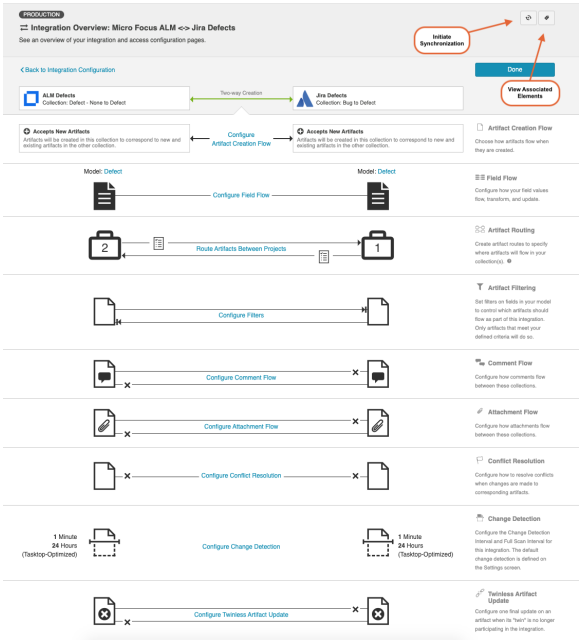



Name and describe your integration, and select your repositories and collections.




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.

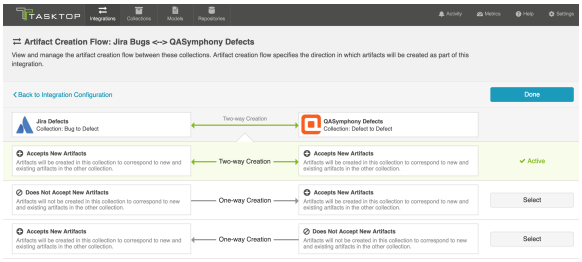
# Artifact Creation Flow

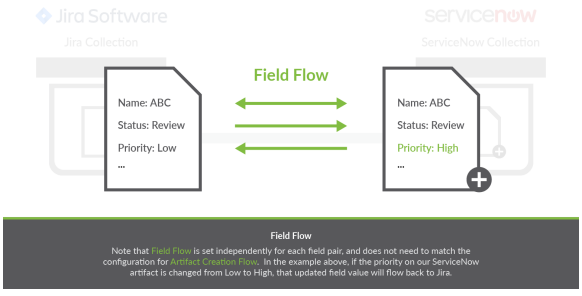
## Introduction

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.

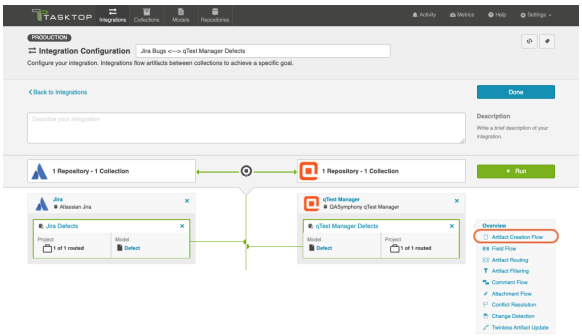
However, once a Jira artifact creates a target artifact in ServiceNow, if any updates are made to fields on the target artifact in ServiceNow, those updated fields **could** flow back over to Jira, based on the integration's *field flow* configuration.  So while the integration is not **creating** new 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.



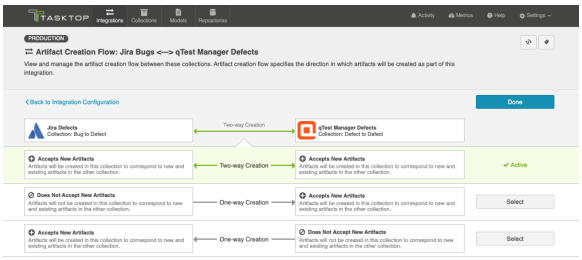




## Configuring Artifact Creation Flow

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**.

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.

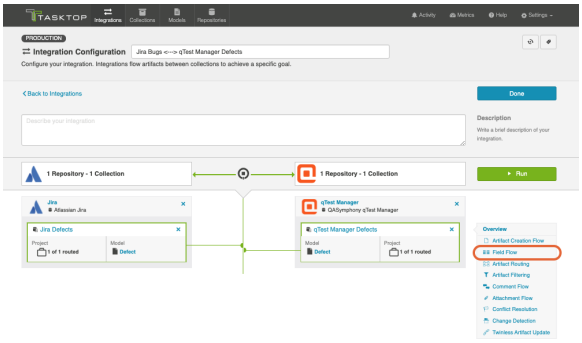
On the Field Flow screen, you can configure:

- 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

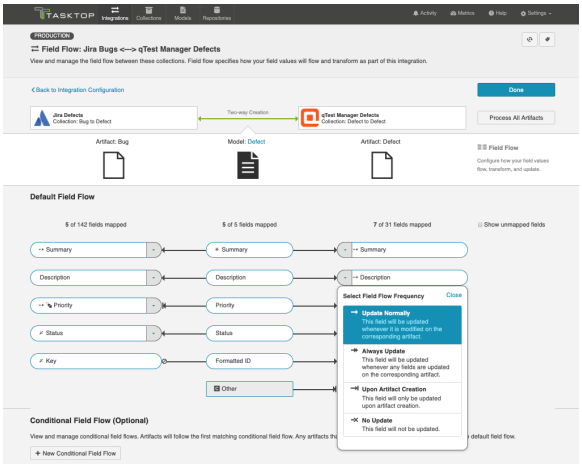
Each field can be configured individually.

## 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**.

# 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)

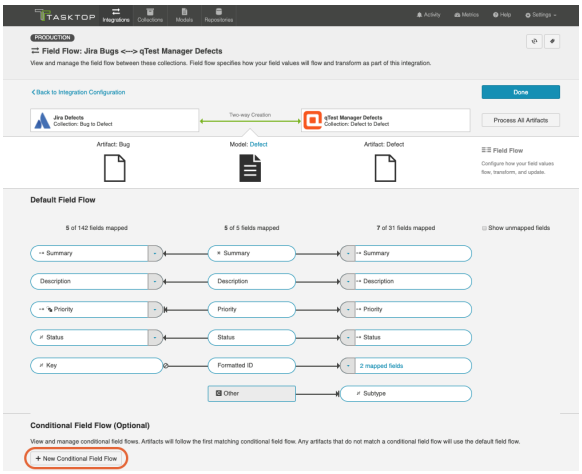
You can create multiple conditional field flows, which will be executed in order from top to bottom.

💡 **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.

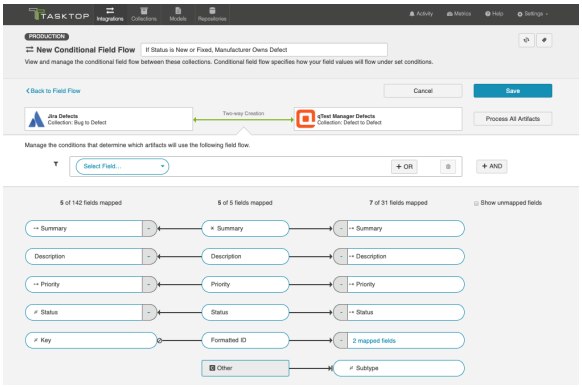
In the scenario configured below, ownership of an artifact changes from the manufacturer (Jira) to the supplier (qTest) when the status of the artifact is updated from **New** to **In Progress**, and back to the manufacturer when updated to **Fixed**. To prevent conflicting data, conditional field flows are used to only flow field values from the owner of the artifact when these changes occur.

## Configuring Conditional Field Flow

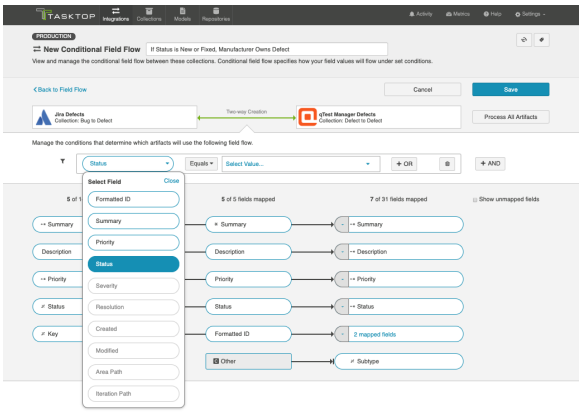
To configure a conditional field flow, click **New Conditional Field Flow**.

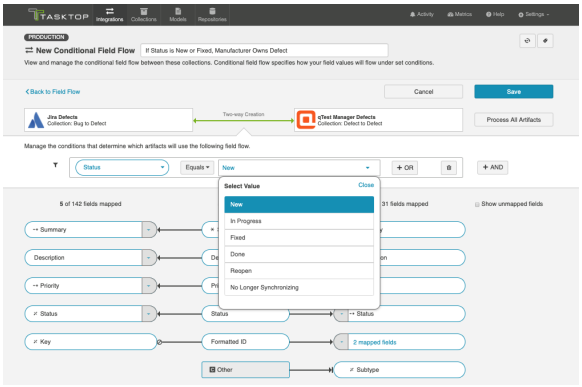


This will bring you to the **Conditional Field Flow** screen.



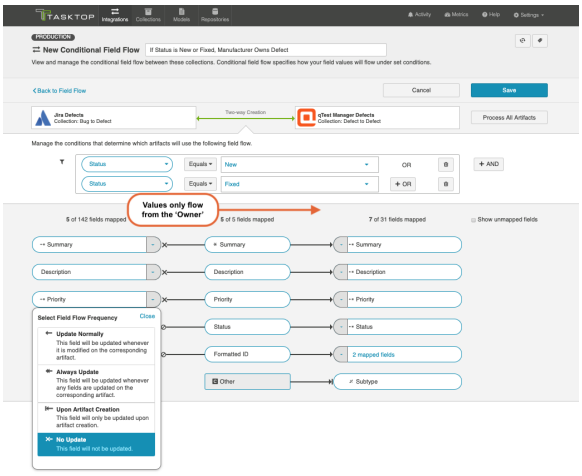

Select a field and set your desired conditions.

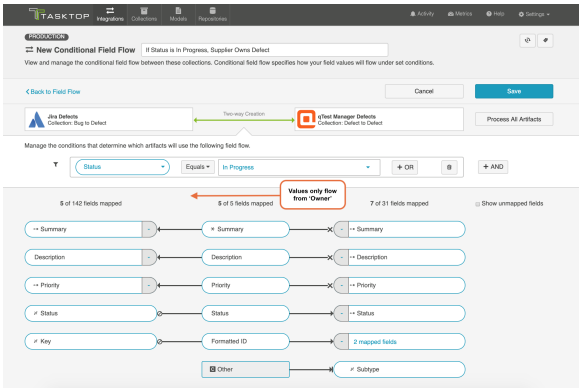


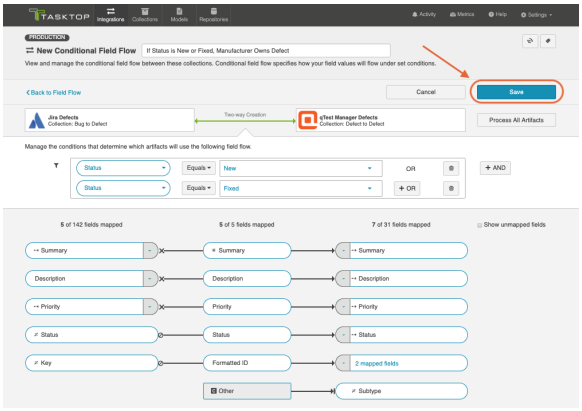

Select your field flow frequency to determine the direction of your field flow.

💡 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).

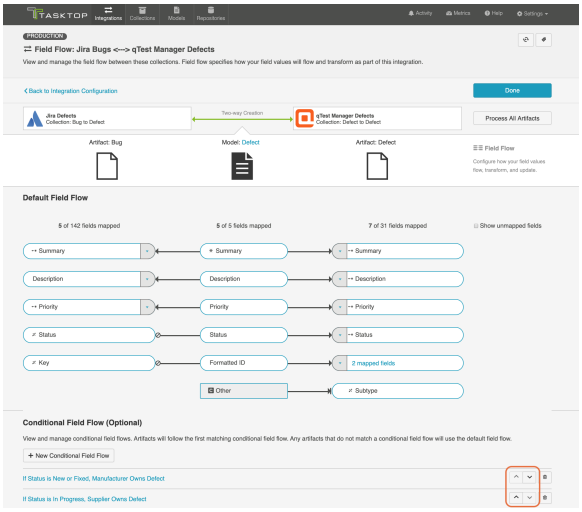



After you have finished configuring your conditional field flow(s), click **Save** and **Done** to save your changes.




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 |
|---|---|
| → | **Update Normally**: This field will be updated whenever it is modified on the corresponding artifact |
| ↠ | **Always Update**: This field will be updated whenever any fields are updated on the corresponding artifact |
| ⇥ | **Upon Artifact Creation**: This field will only be updated upon artifact creation |
| ✕ | **No Update**: 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 |
|---|---|
|  | A constant value will be sent.<br><br>Note that:<br><br>• 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 *other* collection will receive this constant value for the field in question.<br>• 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 *this* collection will receive this constant value for the field in question. |
|  | A state transition will be utilized. Note that:<br><br>• If the icon is on the side of the collection, this means that a state transition graph is being utilized.<br>• If the icon is on the side of the model, this means that a state transition extension is being utilized. You can determine which collection it applies to based on whether it is left-aligned or right-aligned.<br><br>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. |
|  | 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**.

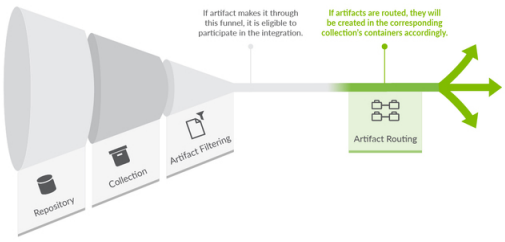
# Next Steps

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.



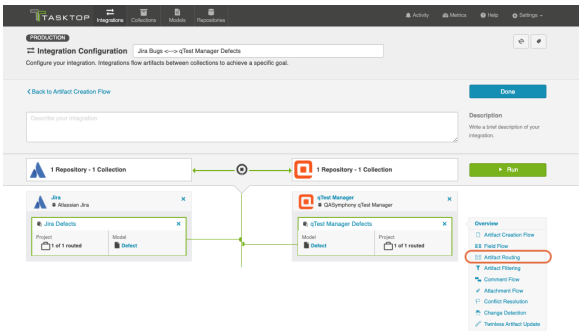

Artifact Routing is needed when artifacts are being created as part of an integration. In addition to knowing the repository in which artifacts should be created, Tasktop also needs to know which container (i.e., project, module, folder, etc) a given artifact should be created in. Specifying the artifact 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.

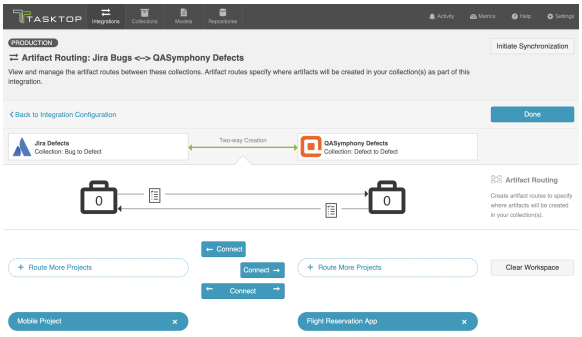


# Static Artifact Routing

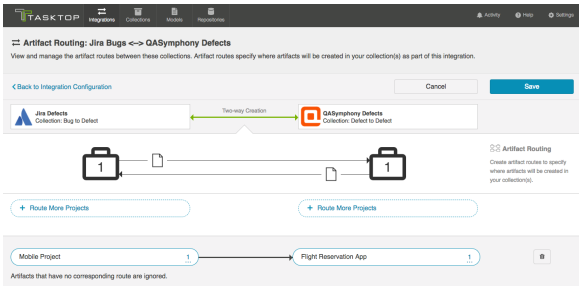
In some cases, the project an artifact resides in in the source collection can sufficiently determine which project an artifact should be created in in the target collection. In these instances, you can configure what is known as 'static artifact routing' (also known as 'explicit artifact routing').

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

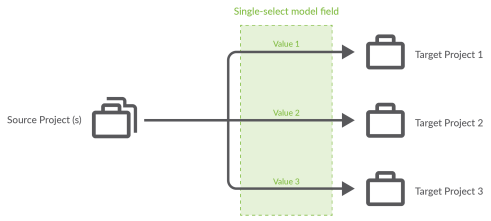
Check out the video below to learn more about Conditional Artifact Routing:

⚠ *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.

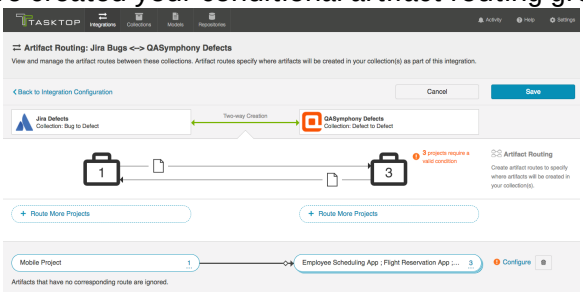
In these instances, you will configure what is known as **conditional artifact routing** to determine which project each artifact is created in within your target repository. Conditional artifact routing (also known as 'dynamic artifact routing') can be used to inspect a single-select field of an artifact 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.

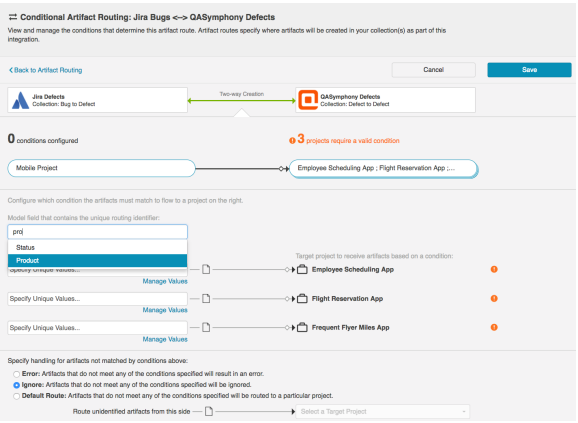
Notice that after you've created your conditional artifact routing group, you'll be prompted to configure your route.



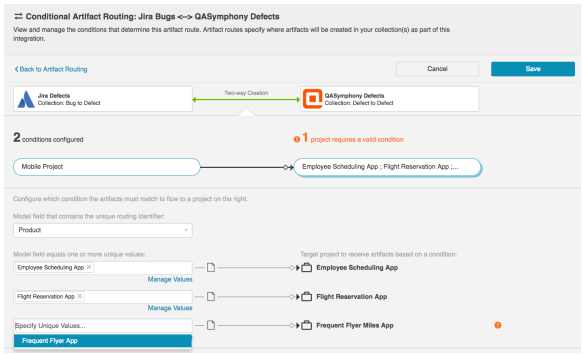

Click **Save** and then click **Configure**. You'll be brought to the Conditional Artifact Routing screen. Here you'll start by selecting the model field that you would like to use to determine your artifact route.

💡 **Note**: Conditional Artifact Routes can only be configured based on **single-select fields** in your model.

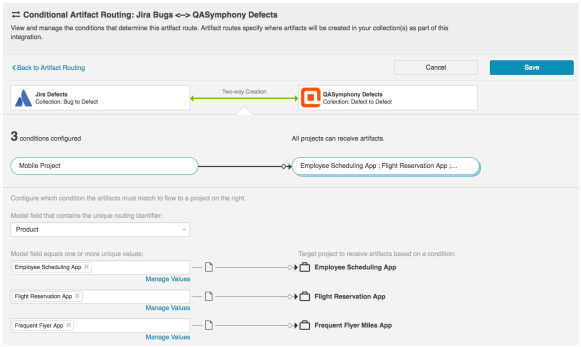
In the example below, the field "Product" contains the unique values that should determine the project an artifact will be created in in QASymphony.




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

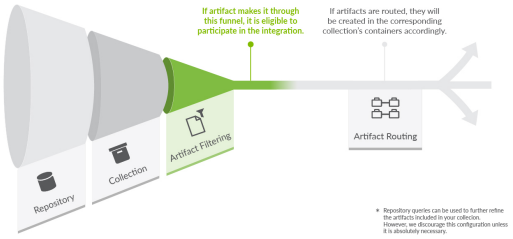
Once you've configured your Artifact Routing configuration, your next step will be to review your Artifact Filtering.

# 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.

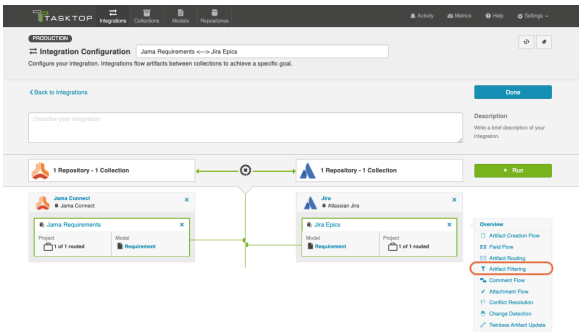
To use a field for artifact filtering, it must:

- 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

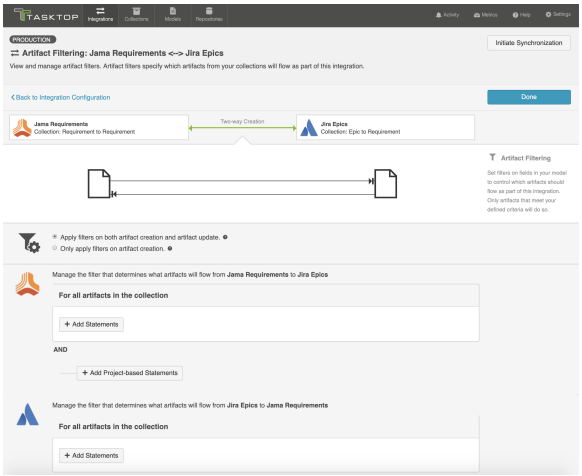
💡 Note that you can utilize our transforms to filter based on an 'unsupported' collection field type, if that field is mapped to a supported field type in your model. For example, you could filter based on a rich text field in your repository, if that rich text field is mapped to a string field in your 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.

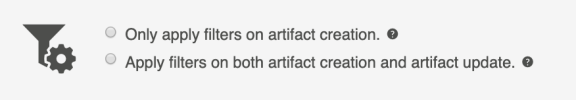



# Artifact Creation vs. Artifact Update

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.




# Configure Artifact Filter Statements

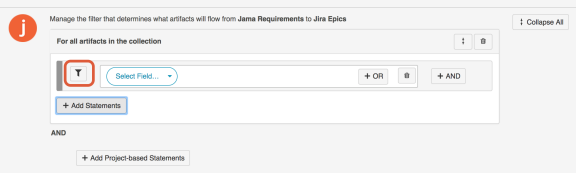
Next, you can begin configuring your artifact filtering statements.  You can add statements for **all artifacts in both collections**, **all artifacts in one collection**, or to **artifacts in specific projects within your collection**.

💡 **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.

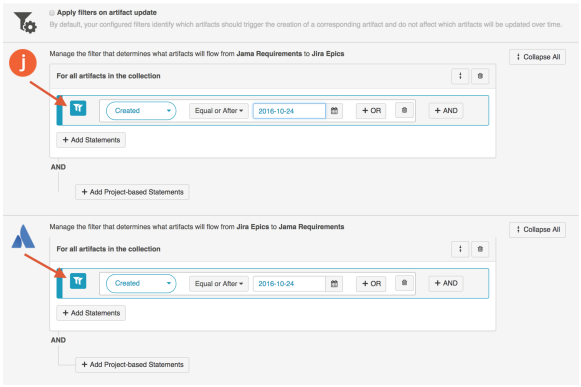

Then, click the filter button. This will apply your filter to the other collection participating in your integration.




You will notice that the button changes to show two filters, indicating that your filter will apply to both collections.

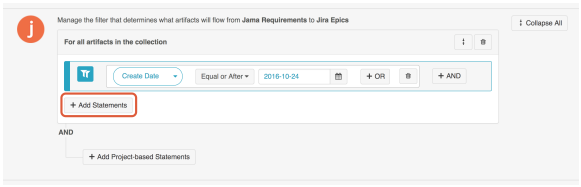
You'll also notice that any modifications you make to that filter statement will automatically be reflected in the other collection. If you'd like to disconnect the filter from both collections, simply click the double-filter button again, and you will be able to edit each filter individually.

Here we are filtering both collections to only create target artifacts that were created on or after October 24th, 2016.

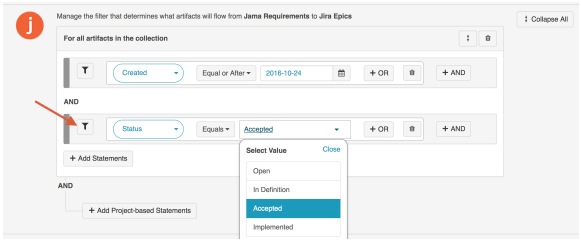



## Apply Filter to All Artifacts in One Collection

To apply a filter to all artifacts in one collection, simply click the **+ Add Statements** button in the desired collection.

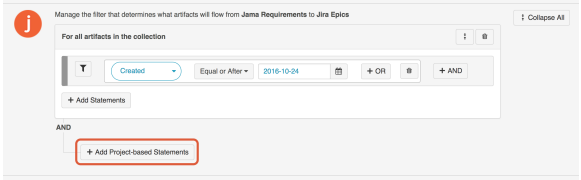



Select your artifact filtering fields and values. You'll see that there is only one filter displayed on the left, which tells you that this filter only applies to one collection in your integration.

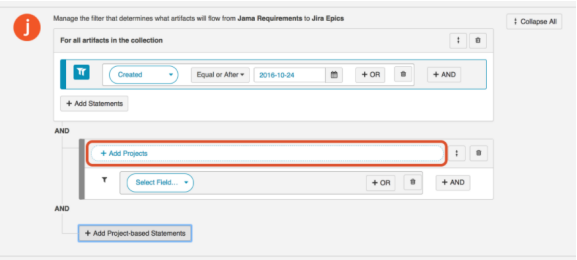



## Apply Filter to Artifacts within Certain Projects in a Collection

To apply a filter to artifacts within certain projects in a collection, click **+ Add Project Based Statements**

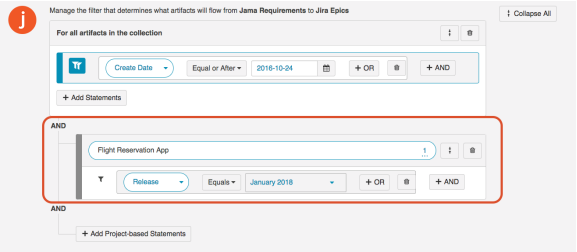



Click **+ Add Projects** to select your project.



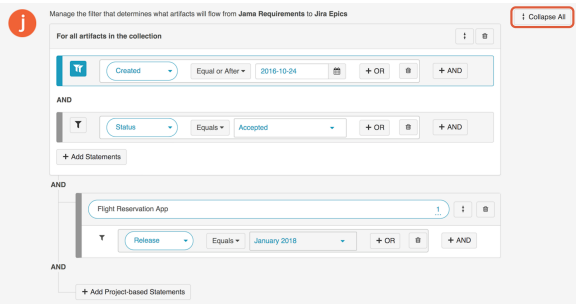

Select the project(s) you'd like your filter to apply to.

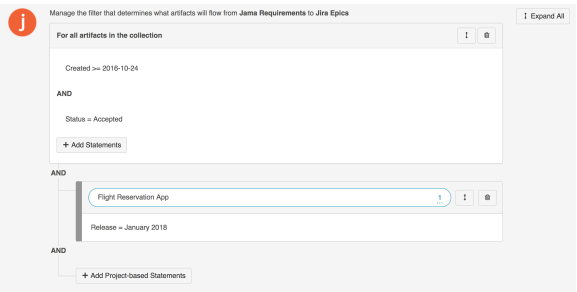
Then click **Select Field...** to begin configuring your filtering statement.




# Viewing Artifact Filter Statements

You can click the **Collapse All** button to view an easy-to-read version of your artifact filtering statements.

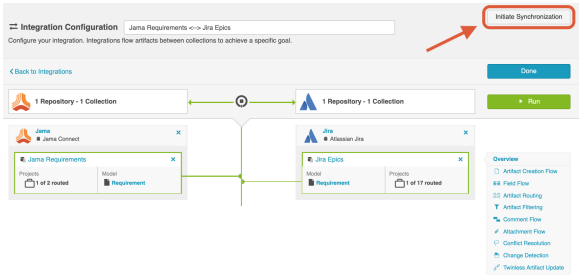


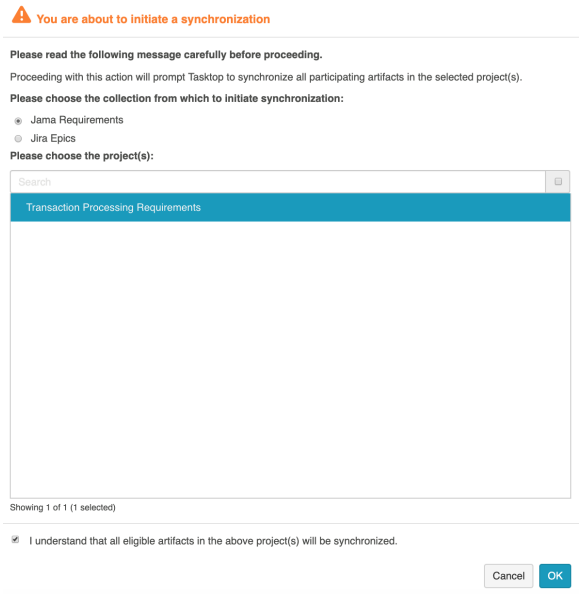

# Expanding Artifact Filters

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.

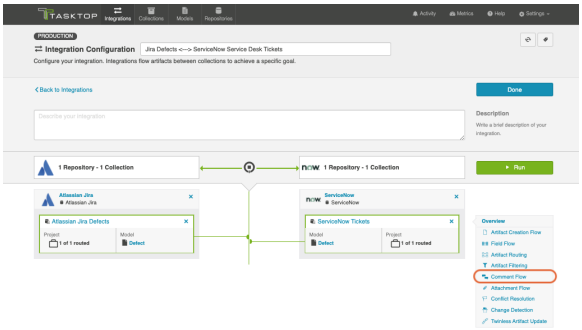
# Comment Flow

## Introduction

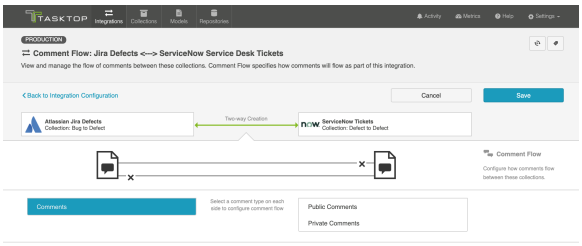
Once you've configured Artifact Filtering, your next step will be to review and update Comment Flow.

## Configuring Comment Flow

To enable and configure Comment Flow, click the **Comment Flow** link on the **Integration Configuration** screen.

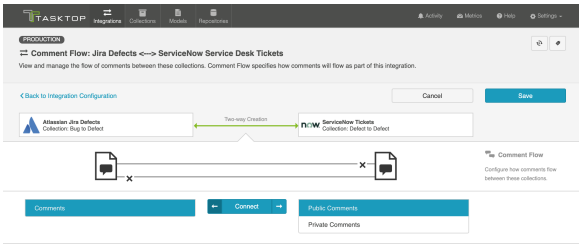


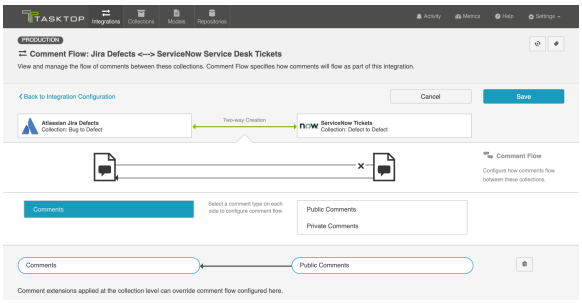

This will bring you to the Comment Flow screen.



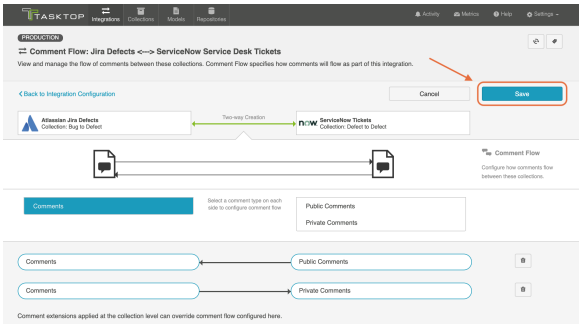

If your collections support comment flow, you will be able to choose your desired Comment Flow style here. You can choose to flow comments bi-directionally or in a single direction.

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.





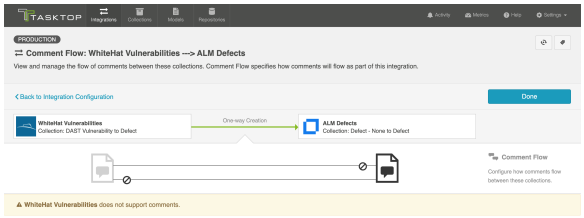

Once you've updated the comment flow settings as desired, click **Save** and **Done** to save your changes.




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 or your collections does not support comment flow, you will see a notice like the one below:



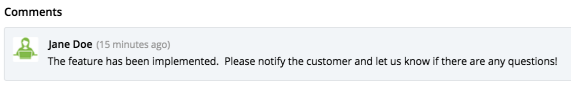

# Comment Impersonation

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,

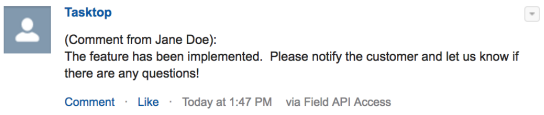
- When your target repository supports impersonation, but Tasktop cannot locate the person with the information provided from the artifact in the source repository,

  **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.

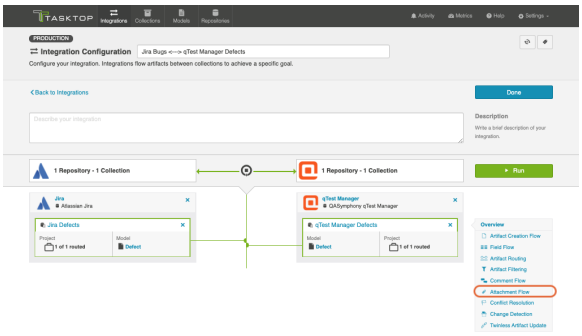
# Attachment Flow

## Introduction

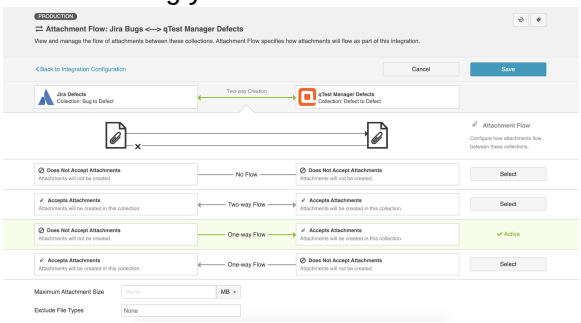
Once you've configured Comment Flow, your next step will be to configure Attachment Flow.

## Configuring Attachment Flow

To enable and configure Attachment Flow, click the **Attachment Flow** link on the **Integration Configuration** screen.



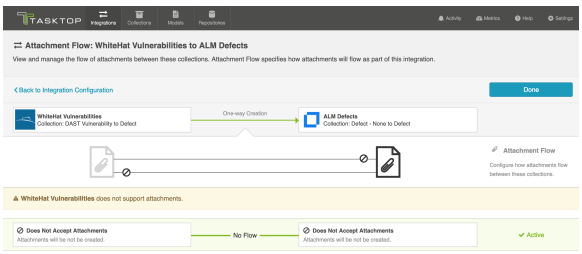

This will bring you to the Attachment Flow screen.




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.

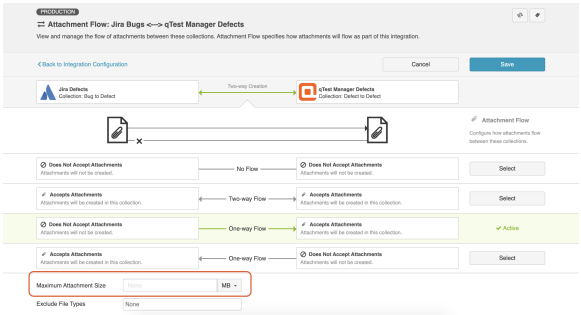
You can check our Connector docs to see which repositories support attachment flow. If one or both or your collections does not support attachment flow, you will see a notice like the one below:



# Maximum Attachment Size

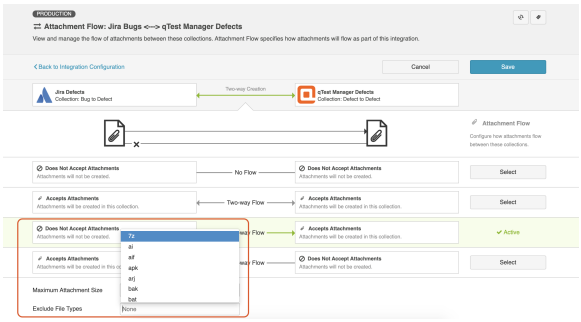
You can also configure the **maximum attachment size**. If attachments are larger than this size, they will be ignored by your integration.

💡 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.



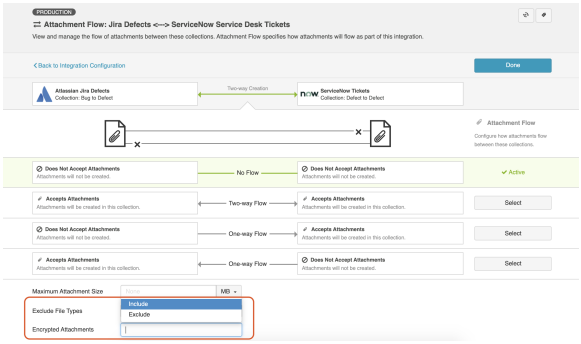

# Exclude File Types

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 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.

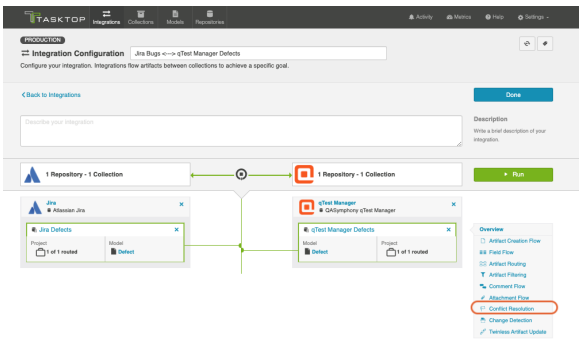
The Conflict Resolution Strategy screen allows you to control how data conflicts will be resolved:

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.

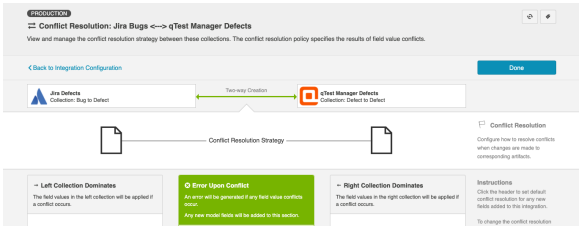
## Configuring Conflict Resolution

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.

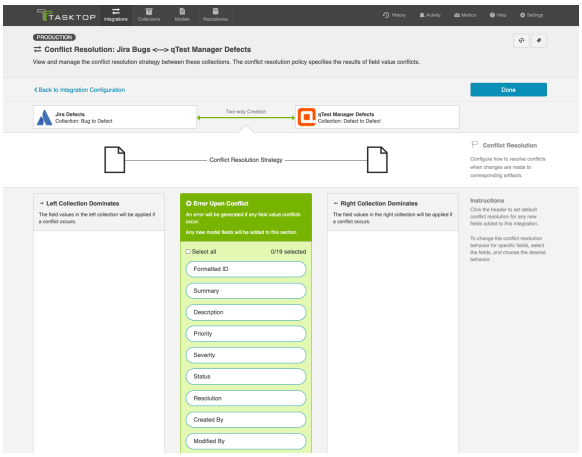
# Field-by-Field Conflict Resolution

*See Tasktop Editions table to determine if your edition supports Field-by-Field Conflict Resolution functionality.*

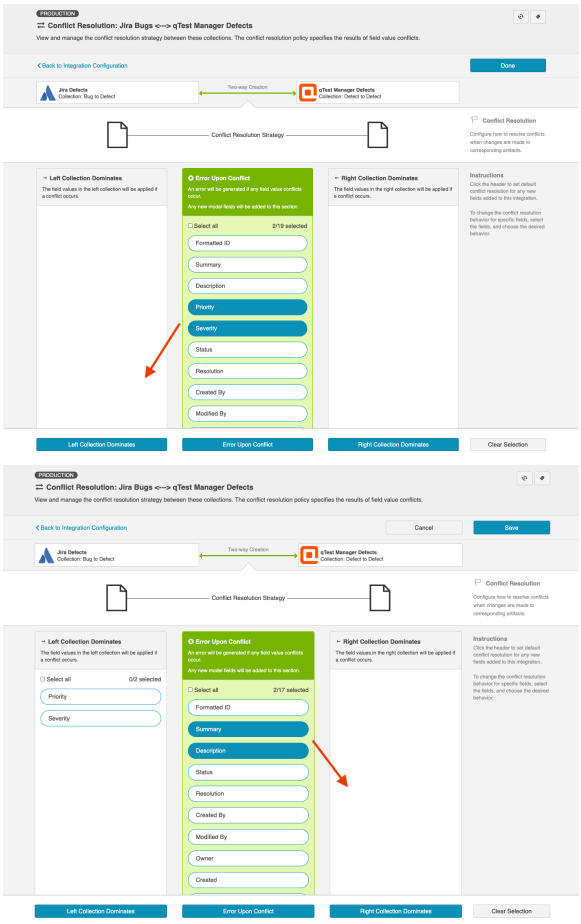
You can also configure your conflict resolution strategy on a field-by-field basis.

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.



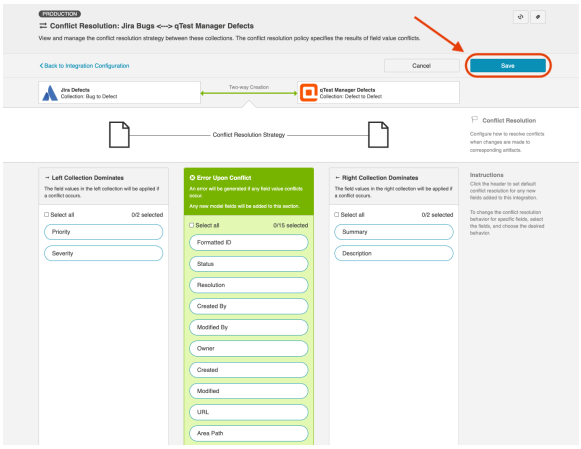

As you can see in the scenario below, the user has chosen their Jira collection to dominate for the field values **Priority** and **Status** and their qTest collection to dominate for the field values **Summary** and **Description**.




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.



# 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. However, if you'd like to override the global defaults, you can configure integration-specific change detection and full scan intervals on this 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.
  - ⚠️ For Cloud instances of Hub, we recommend setting the change detection interval to at least 1 minute.
- 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.
  - ⚠️ For Cloud instances of Hub, we recommend setting the full scan interval to at least 24 hours.

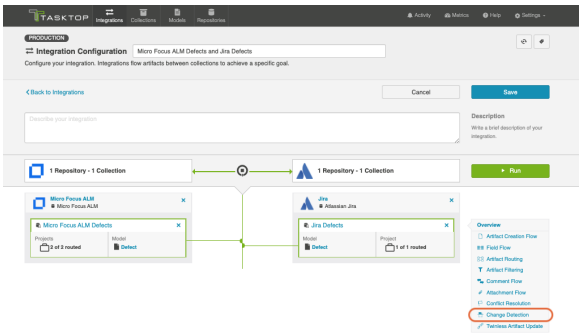
Not all changes to an artifact will register as a change. Some repositories do not mark items as changed when (for example) a relationship is added or an attachment has changed. These may not be picked up by regular Change Detection, but will be picked up by a Full Scan. You can review our connector docs t o see types of updates that will require a full scan. The Full Scan Interval defaults to 24 hours on the Ge neral (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.

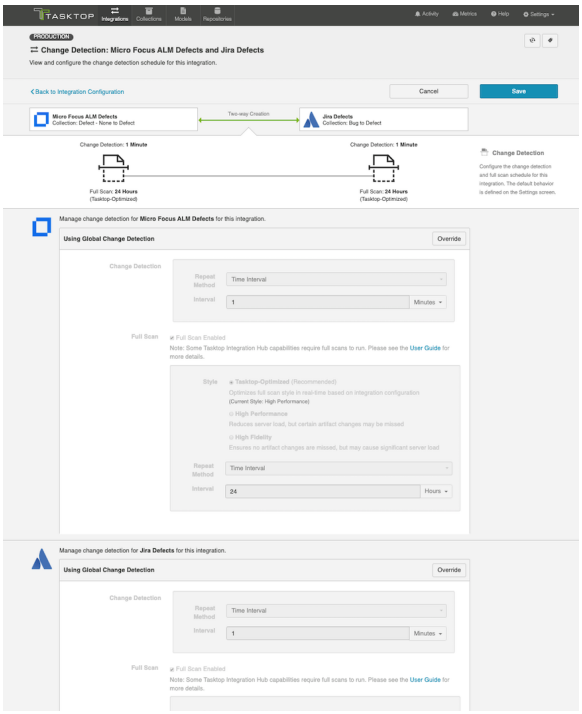
You can learn more about change detection and full scan styles in our FAQ here.

## Configuring Change Detection

To configure integration-specific change detection, click the **Change Detection** link on the **Integration Configuration** screen.

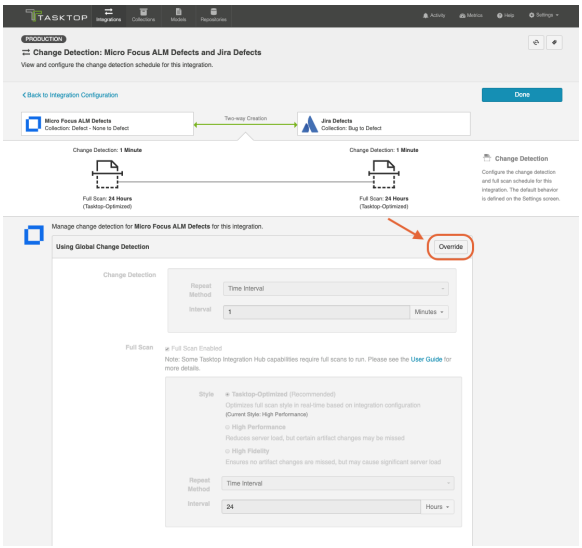


This will bring you to the Change Detection screen, where you can view the current change detection and full scan intervals configured for each collection in this integration. These will default to the global intervals configured on the General (Settings) screen.

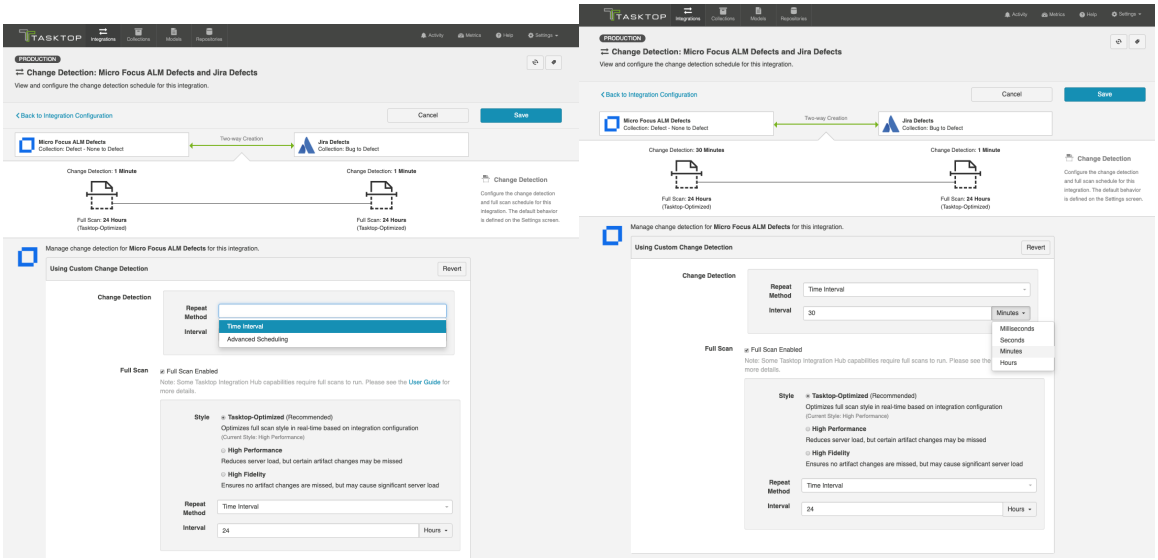


To override the current settings, click the **Override** button on your desired collection. This will allow you to set a custom change detection and/or full scan interval for each collection within the context of this integration.

💡 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:



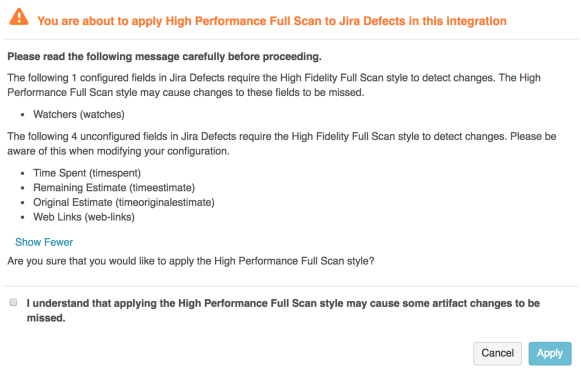
# Full Scan Style and Interval

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.

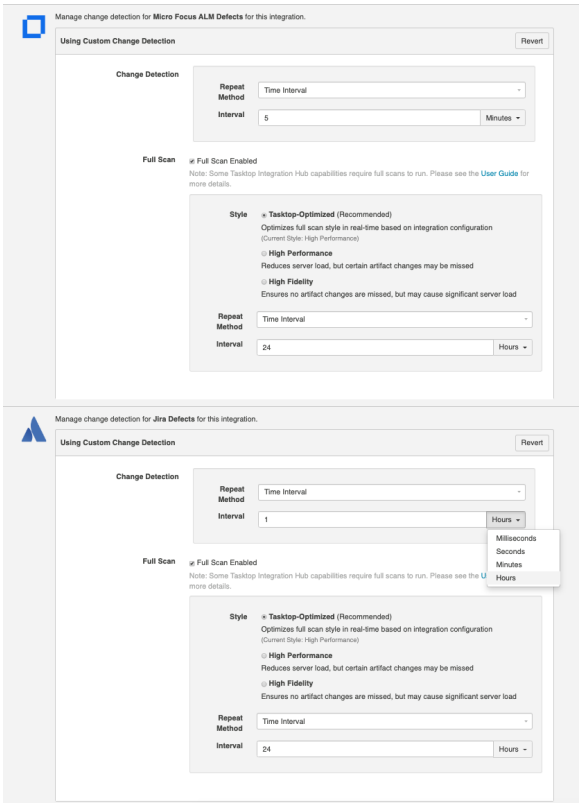
The following full scan styles are available:

- **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.

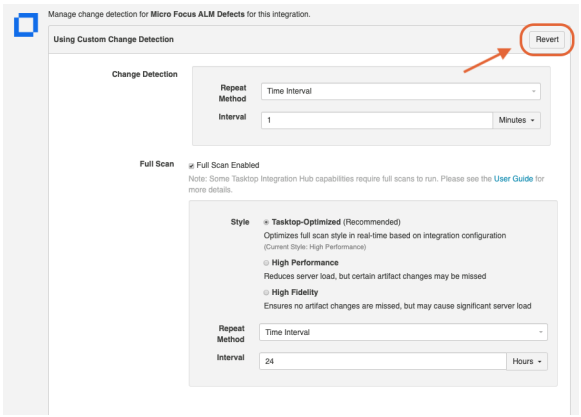
If High Performance style is selected, Tasktop will provide a warning identifying any specific artifact changes which may be missed:



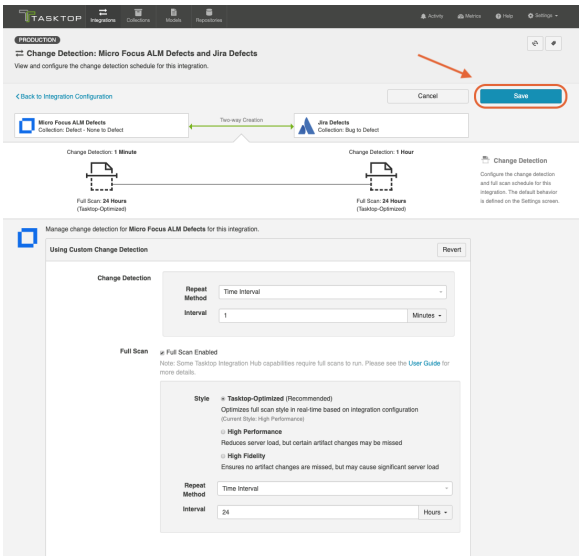

Once you have configured your change detection and full scan intervals for your first collection, you can update the settings for the remaining collection, if desired.




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.

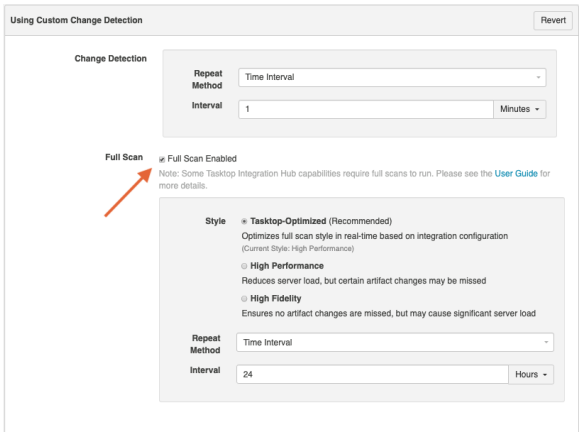



## Disabling Full Scan

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

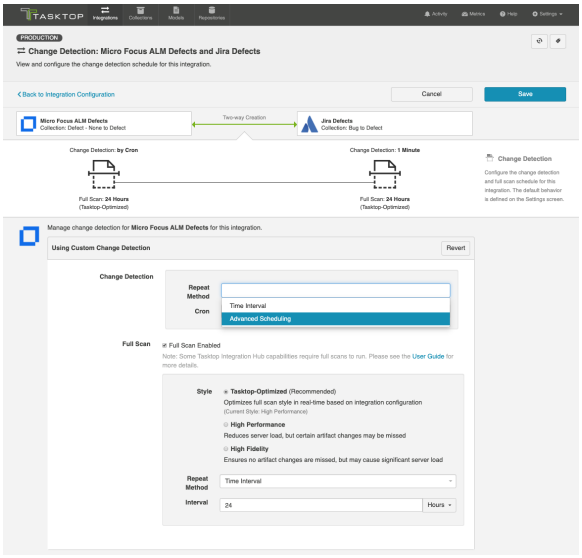
*Based on your edition, you may have the ability to configure change detection using a Cron Expression. To do so, please see the section below.*

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.

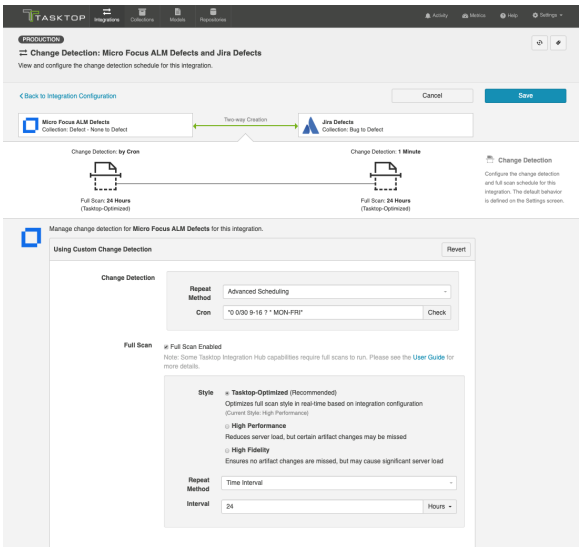
💡 **Note**: Cron expressions must be written using the Quartz cron format to be valid.

Learn more in our FAQ here.

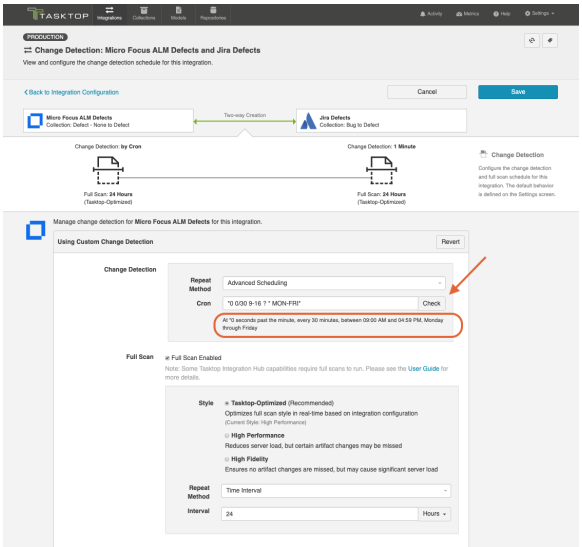
To utilize cron expression for your full scan or change detection, select **Advanced Scheduling** in the Repeat Method field.

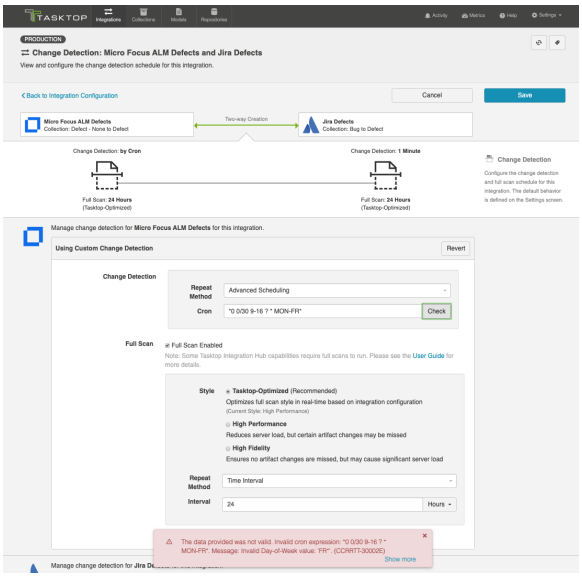


You can then enter the cron expression for your desired scheduling. For example, if you would like to run a full scan every 30 minutes from 9am to 5pm from Monday through Friday, it would be written as follows: *0 0/30 9-16 ? * MON-FRI*




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:

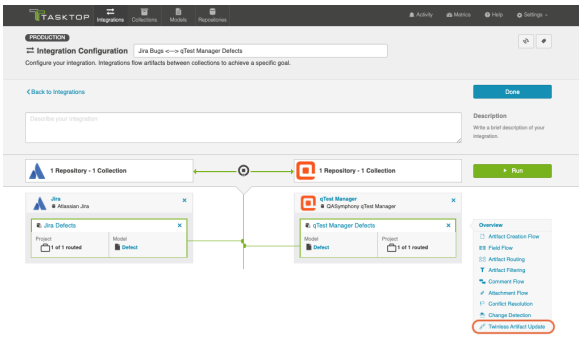
1. Artifacts fall out of Artifact Routing. This can happen when:
    a. Artifacts are deleted
    b. Artifacts are moved to projects not routed as part of the integration
2. Artifacts no longer meet the Artifact Filtering criteria.

💡 **Note**: If an artifact leaves the integration due to a repository query, twinless artifact update may not trigger. Contact Tasktop Support for more details.

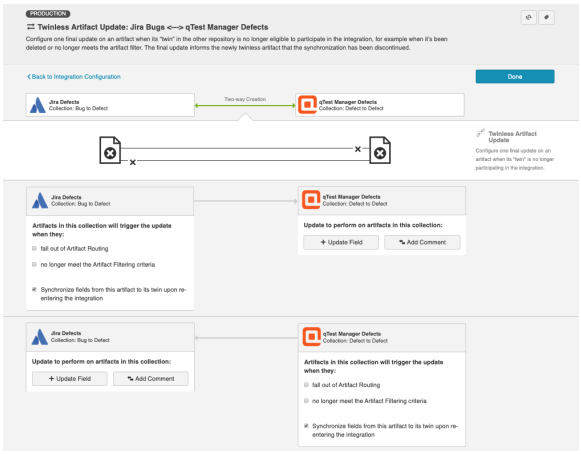
You can configure the update to occur based on one or both of the criteria above, as well as the type of update made to the 'twin' artifact. For example, you could add a comment saying "Artifact is no longer synchronizing" or change the state of the twin to "No Longer Synchronizing" or clear out a a 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.

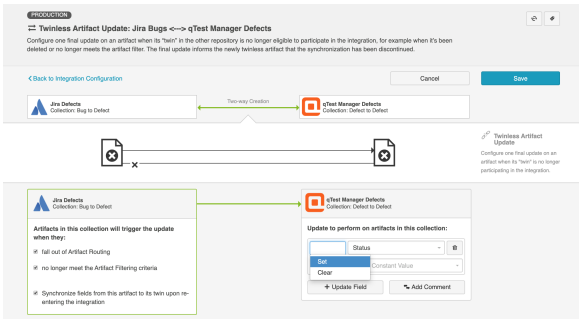
Depending on the type of update and the repositories used, it may take until the next High Performance Full Scan for Tasktop to detect that an artifact is no longer eligible for the integration.

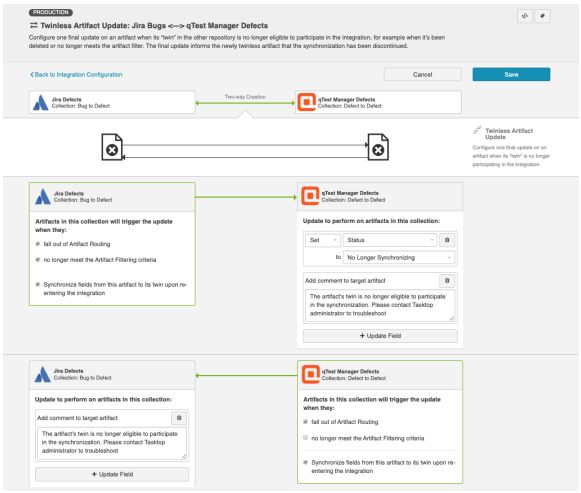
If you are **updating a field**,

- 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




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**.

# 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.

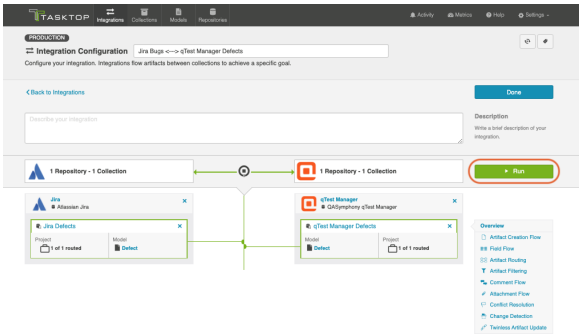
Tasktop does *not* support deletion of artifacts across repositories. See this FAQ for additional information.

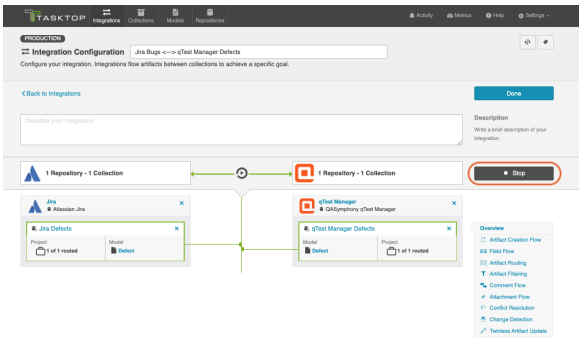
### Running your Integration

There are two ways to start or stop your integration:

## From the Integration Configuration Screen

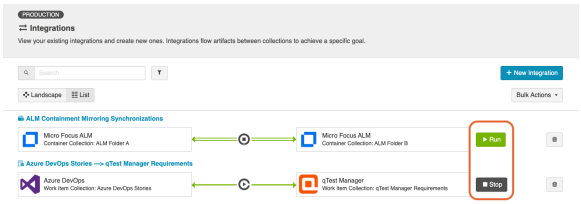
Simply click **Run** to run the integration, and **Stop** to stop the integration.

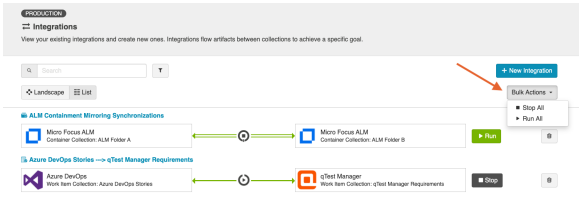



## From the Integrations List Screen

Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** b utton to run or stop all integrations.




# Next Steps

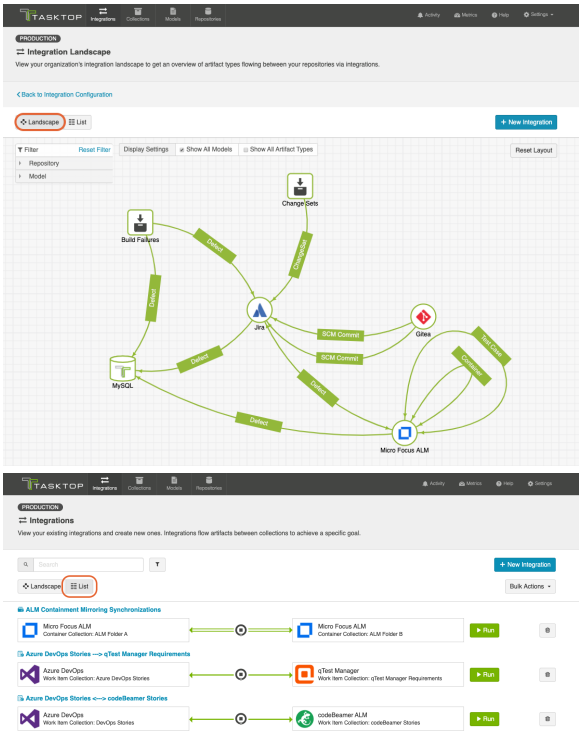
You can learn how to view and visualize your integrations here.

# 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.



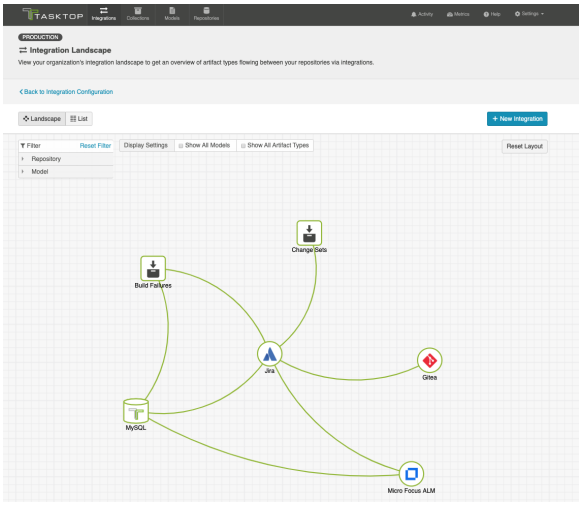

## Landscape View

*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:

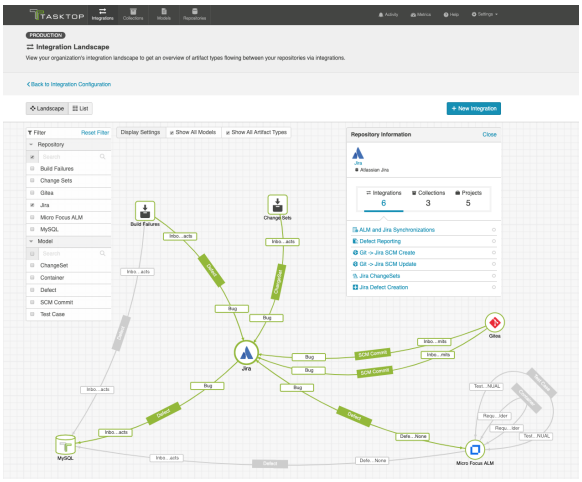



If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

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:

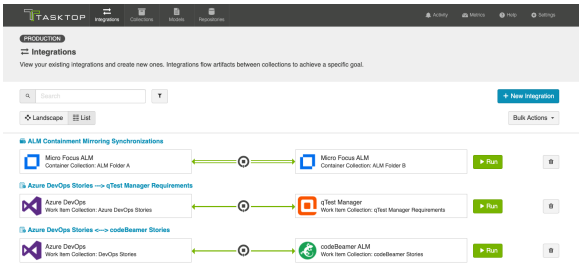



# List View

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.

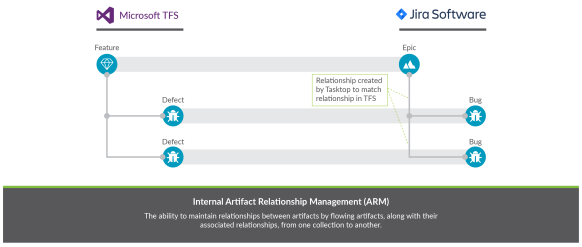
# Synchronizing Relationships

## 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

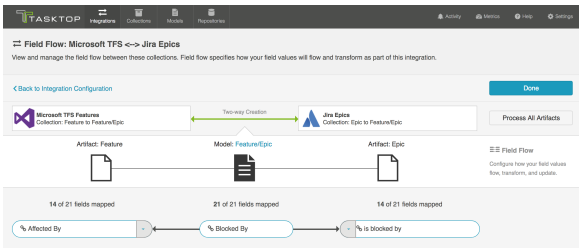
Below, we'll outline an **Internal ARM** scenario where we flow Microsoft TFS features to Jira epics, in addition to the defects that block them, all while preserving the relationships between the artifacts within each internal system.




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.

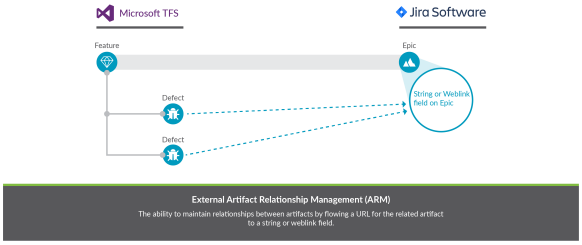



3. Next, configure your Defect  Defect Synchronize Integration as you normally would.
4. Run both integrations. You will see your epics and features, and your defects, as well as *their relationships to one another* successfully flow as part of your integration.

💡 **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

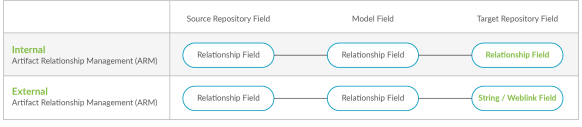
If you'd like a more lightweight approach, you can configure the scenario below to flow the URL of the related artifact in the source repository to a weblink or string field in the target repository. This is what we refer to as **External ARM** (Artifact Relationship Management).




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.




To configure External ARM in Tasktop, follow the instructions below:

💡 First, confirm that both repositories support the following in our Connector Documentation:

For the source repository:

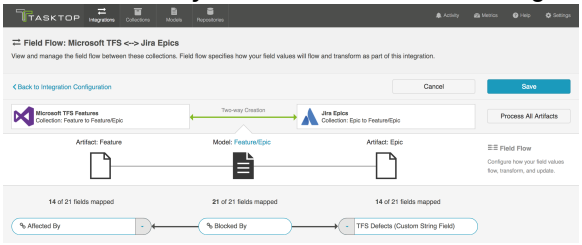
- Relationship field types are supported
- The related artifact type (whose URL you would like to flow) is supported, and provides a unique URL

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** t arget 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:

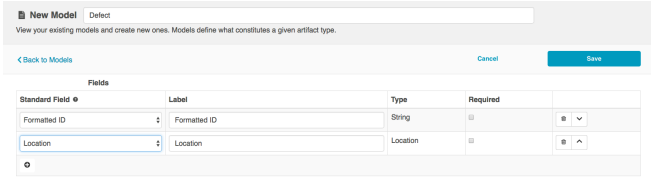
- You will need to specify which field to pull the source artifact's ID (or URL) from
- You will need to specify which field to use to store the source artifact's ID (or URL), in your target repository


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:

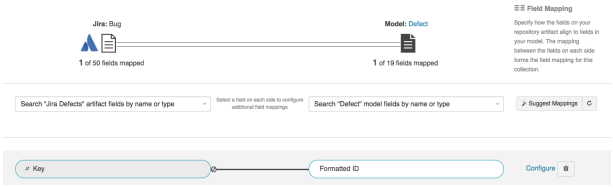
1. Go to the **Model** that you are utilizing in the integration.  Ensure that your model includes the Formatted ID field.

   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.

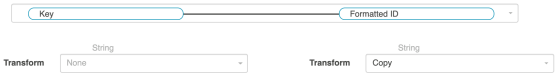



2. Go to the **Collections** screen for *each* of your repositories, and set up mapping to tell the integration where to pull the ID from:

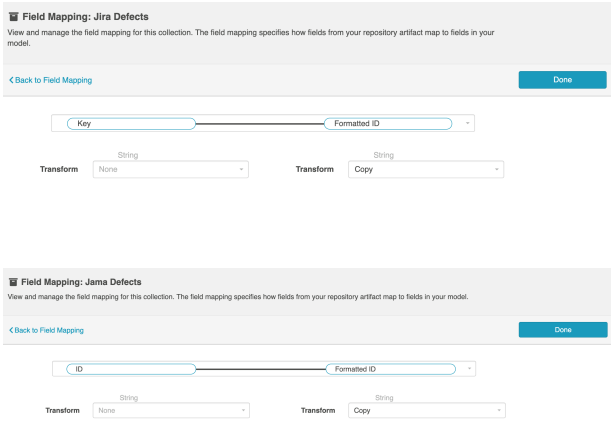
a. Map the Formatted ID model field to the corresponding field in your repository. This is the field that the collection will take the ID data from. Note that Formatted ID is called 'Key' in Jira, but may be referred to using a different name in a different repository (i.e. 'issue ID')




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.

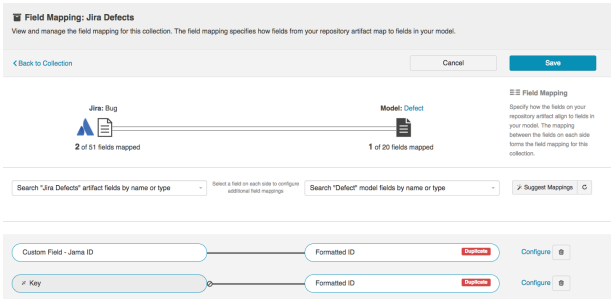



c. Repeat these steps in your other repository.
d. Here is how the mappings should look in each repository, for your *source* fields:




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**:

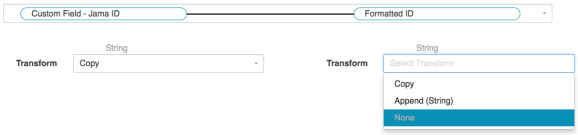
a. Navigate to one of your **Collections.**
b. Map the Formatted ID model field to your repository once more, this time to determine where you would like to **store** this data in your target repository. The field mapping page will tell you that this is a 'duplicate,' but that is ok!




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.

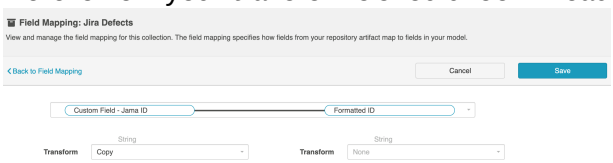
💡 Note: Do not click 'Save' yet. If you do, you will get an error. Continue to the next step below.

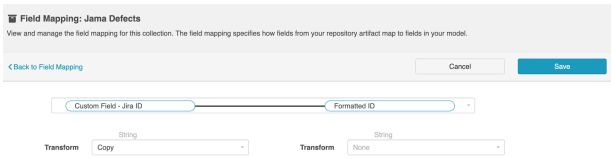
c. Click 'Configure' on the new mapping, and configure as shown below. This will tell the collection to take data from the model and send it to the 'Description' field, but not vice versa.



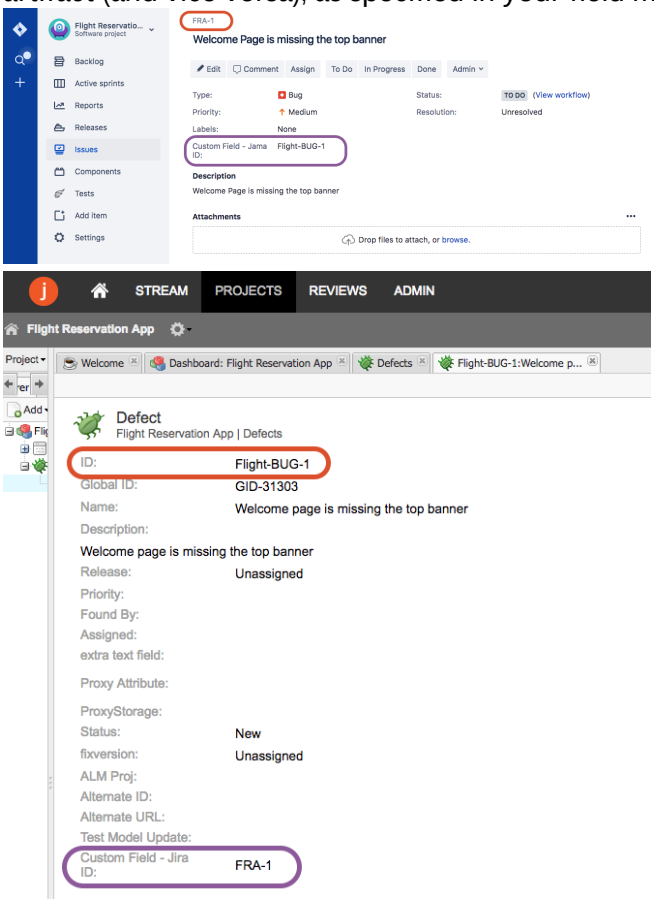

💡 Note: The transform on the left may be 'Copy,' 'Formatted String to Rich Text,' or some other transform depending on the field types of the repository field and model field. However, the important thing is that the transform on the right (on the model side) be set to 'None.' This ensures that data 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:



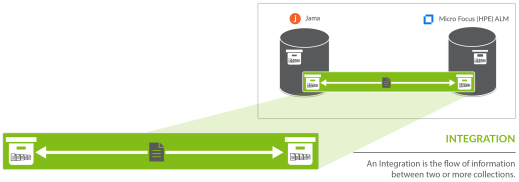



4. When you run the integration, the ID of the source artifact will now flow to a field on the target artifact (and vice versa), as specified in your field mapping:



# Container + Work Item Synchronization

## What is an Integration?



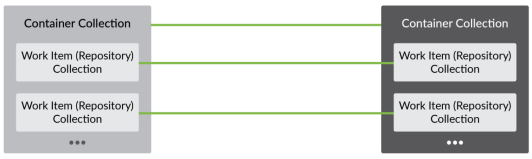

An *integration* is quite simply **the flow of information between two or more collections**. When you configure your integration, you can customize the field flow, artifact routing, artifact filtering, as well as enable or disable comment flow or attachment flow.

## 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.

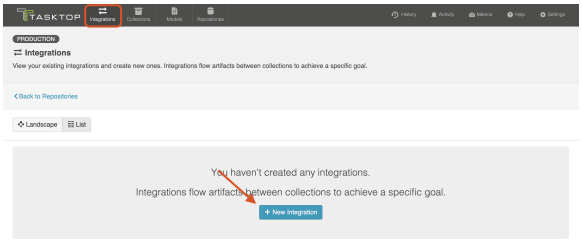



## Configuring a Container + Work Item Synchronization Integration

Once you have your base repositories and collections set up, you can configure integrations to synchronize the artifacts in those collections.

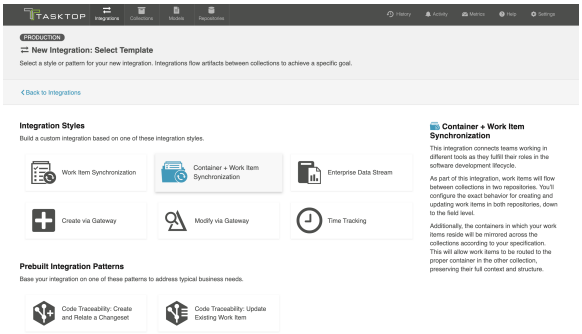
In this scenario, we'll show you how to configure an integration that flows containers (folders) along with the work items (requirements) contained within them, from a source repository to a target repository.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

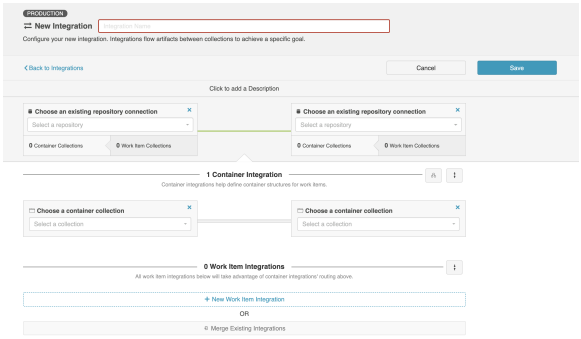



Select the **Container + Work Item Synchronization** integration template.

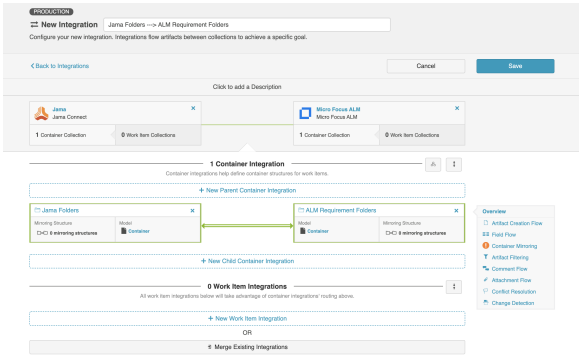
💡 Depending on the edition of Tasktop you are utilizing, you may not have all options shown here.



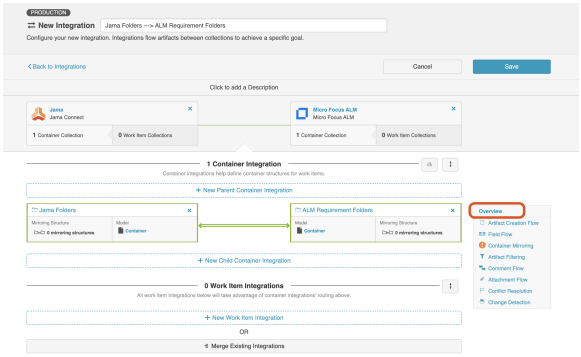

This will bring you to the **New Integration** screen:




Name your integration and select your repositories and container collections, and then click **Save**.




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.

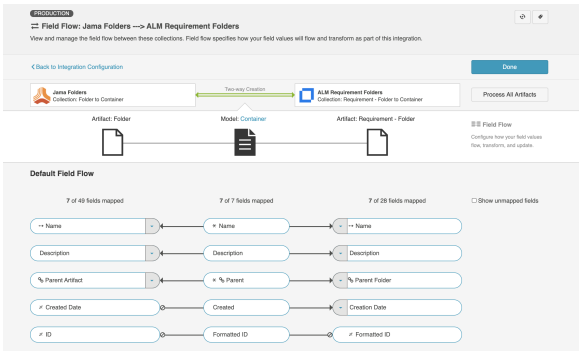
Configuring your Container Integration is very similar to configuring a Work Item Synchronization. Please refer to that page for details, while taking note of the key differences outlined below.

## Artifact Creation Flow

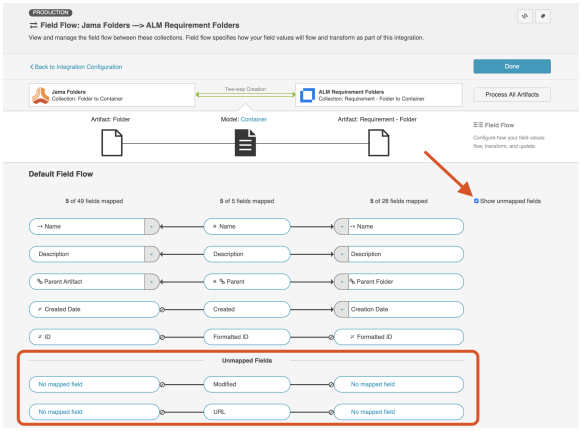
This process is the same as it is for a Work Item Synchronization. Refer to the Artifact Creation Flow page for details.

## Field Flow

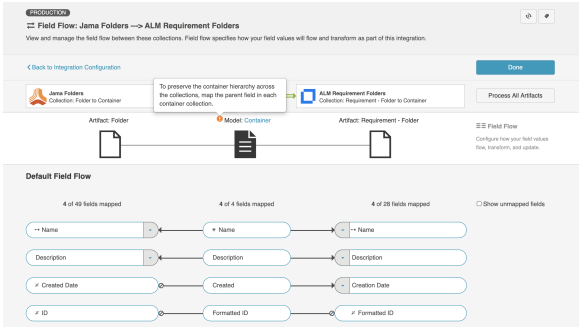
Similar to a Work Item Synchronization, you can click **Field Flow** to configure how fields will flow in your Container Integration. Typically, container integrations will flow significantly fewer fields than a work item integration.



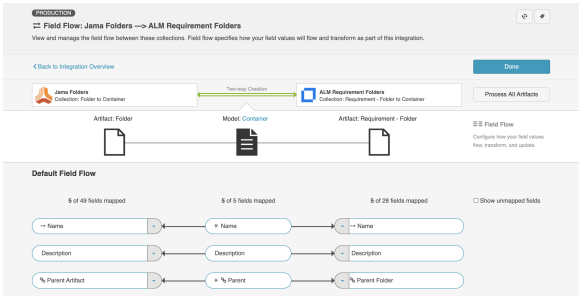

From the Field Flow screen, you can see the names of the mapped repository fields for each collection on the far left and right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by toggling the **Show unmapped 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.



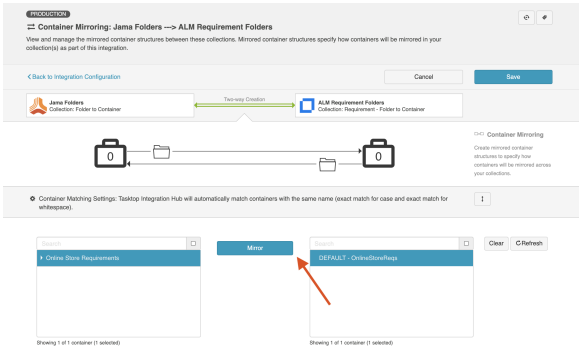

# Container Mirroring

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.

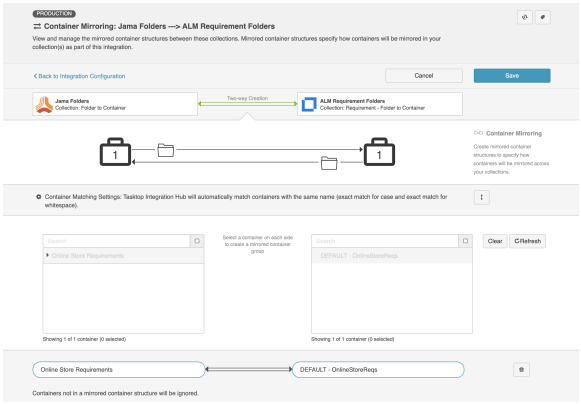
💡 Note that the container structure underneath the top level container will not display in Tasktop unless those container levels can also serve as 'top level containers' for the purposes of mirroring.

Unlike Artifact Routing, Container Mirroring pairs must be one-to-one.

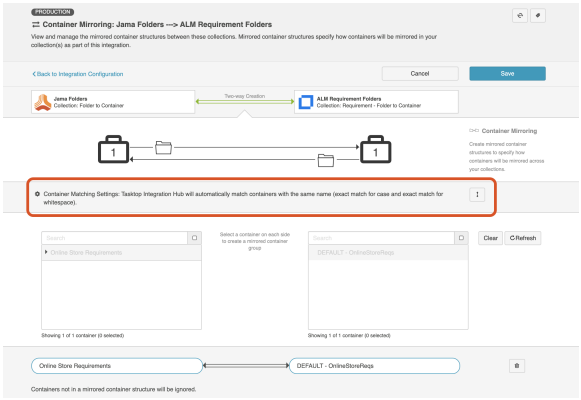



In the example above, any folders contained within the Online Store Requirements project in Jama will create corresponding folders in the Online Store Requirements project in Micro Focus ALM, and vice versa.

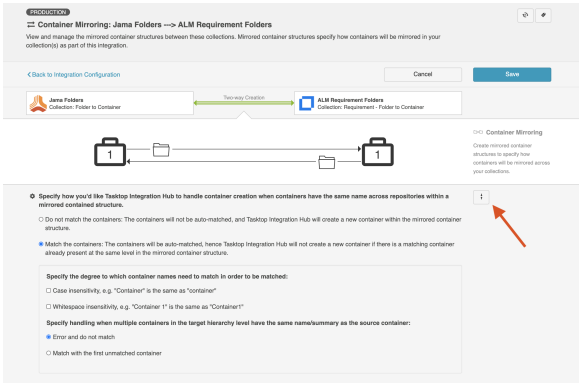
Once you've completed mapping your mirrored pairs, you'll see them in the grey sash below:




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.

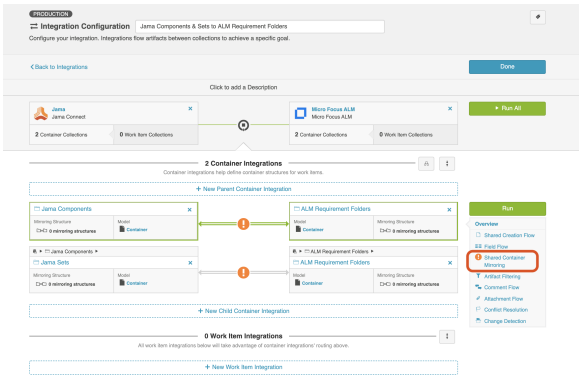
When configuring a new integration, the container matching settings will default to **match the containers** with **error and do not match** selected.

## 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.

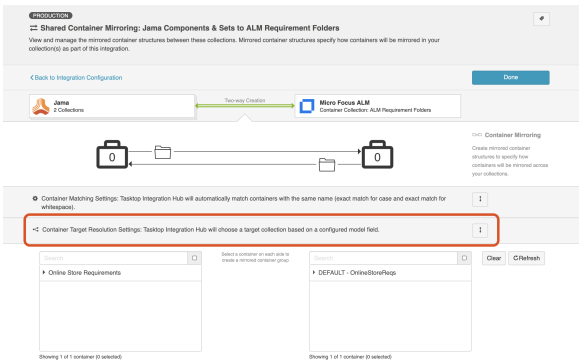
To configure shared container mirroring, set up a container integration where the same collection type is reused in multiple levels of the integration.

You'll then see the **Shared Container Mirroring** link on the **Integration Configuration** screen.



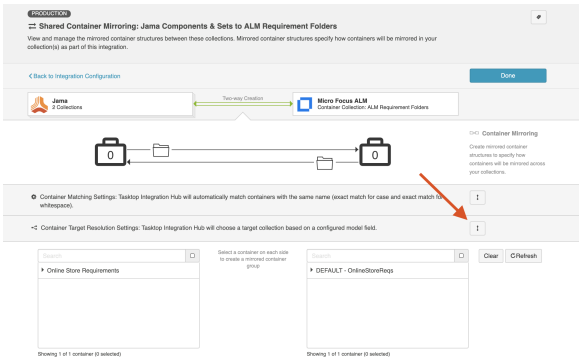

The **Shared Container Mirroring** screen is similar to the Container Mirroring screen, with the exception of **Container Target Resolution Settings**. This setting will appear if artifact creation flow is configured to flow *away* from containers sharing a type.

💡 **Note**: If this setting appears on the Shared Container Mirroring, it **must** be configured for artifacts to flow in your integration.



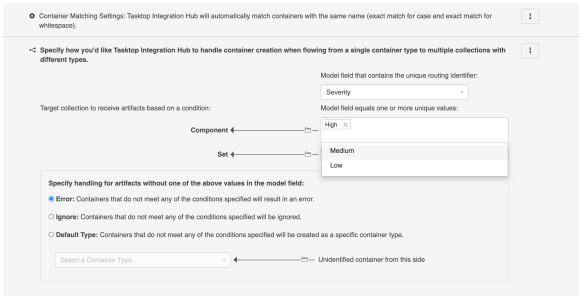
Click the **expand** button to configure your **Container Target Resolution** settings.

💡 **Note**: When configuring a new integration, the container target resolution settings will default to **error** unless otherwise specified.

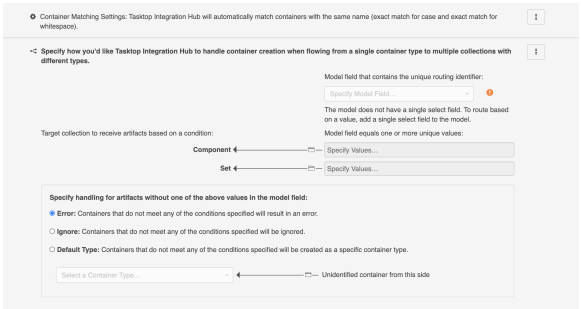



To configure container target resolution settings, select a model field and choose which collection to flow into based on values for the selected field.

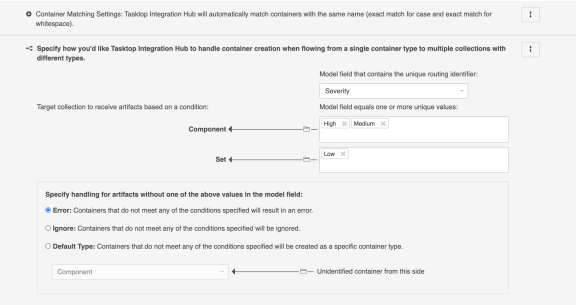
⚠️ **Note**: The model field must be a single select field and must have options configured in the model. The selected model field must also be mapped bidirectionally.




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.

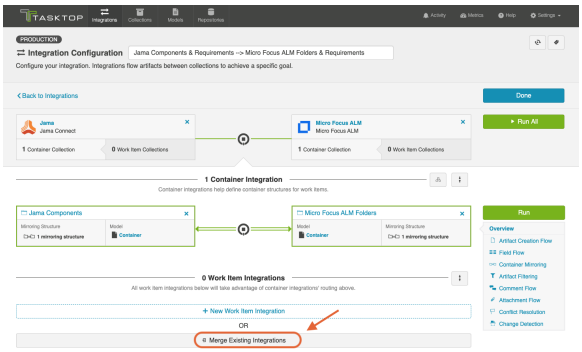
# Configuring your Work Item Integration(s)

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

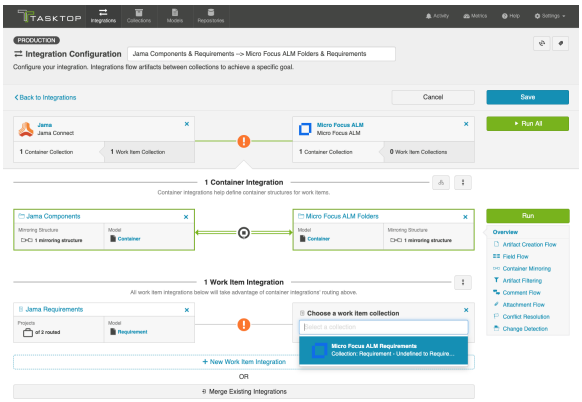
To create a new Work Item Integration, click **+ New Work Item Integration**.




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.

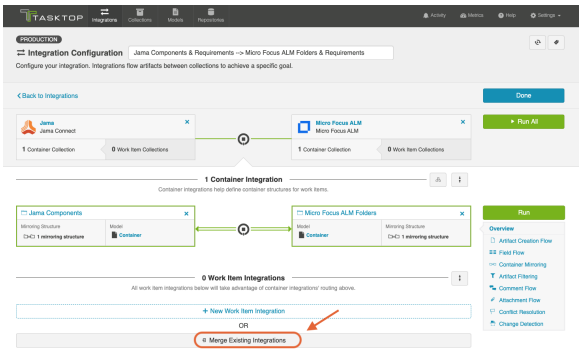
## Merging an Existing Work Item Integration

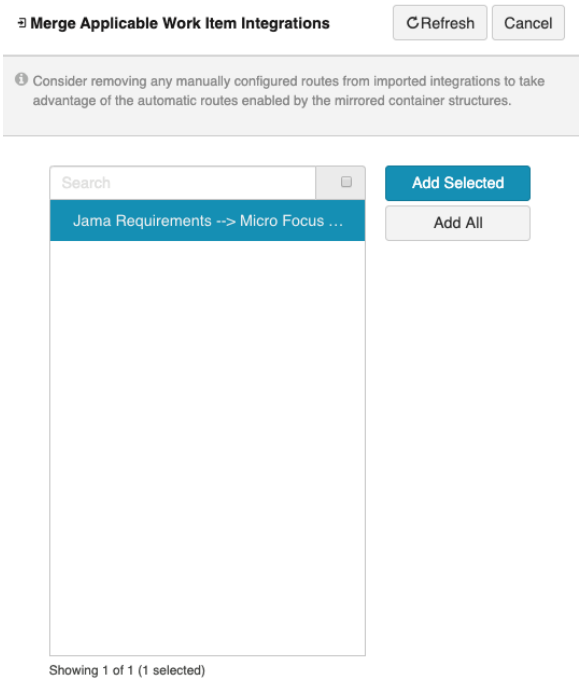
If you've already configured a Work Item Synchronization that you'd like to run as part of this integration, you can add it by clicking **Merge Existing Integrations**.

⚠️ 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).





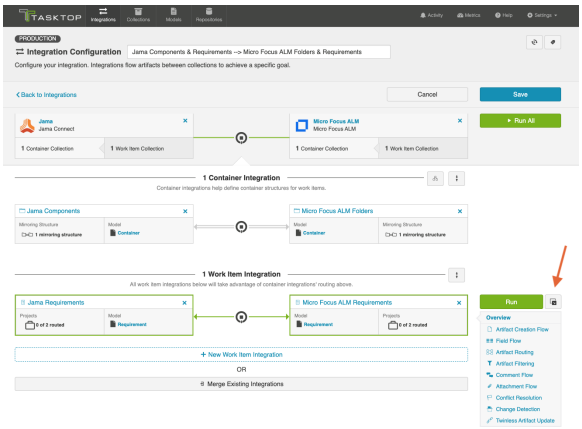

When merging an existing integration, consider removing any manually configured routes from that imported integration to allow it to take advantage of the automatic routes enabled by the mirrored container structures.

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.

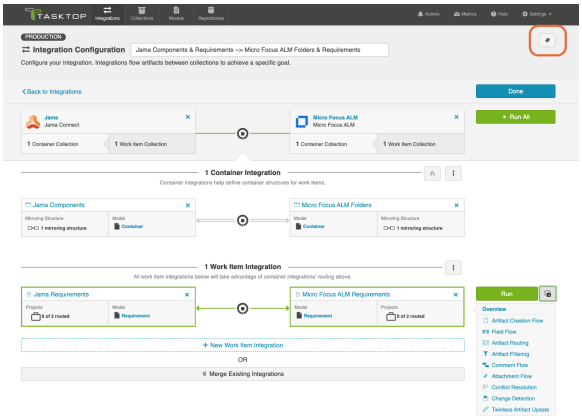
## Activating the Configuration Pane

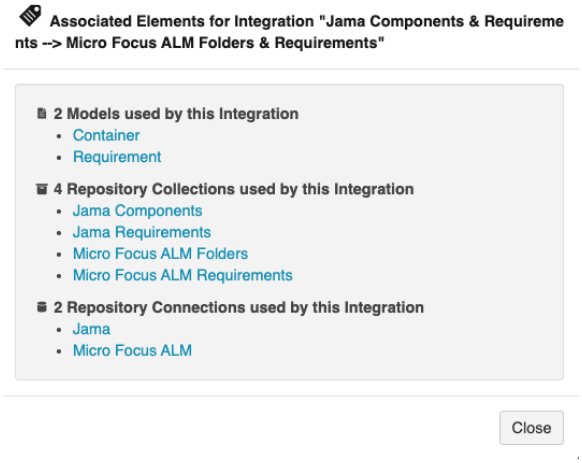
To activate the configuration pane for the integration you'd like to modify, highlight the integration by clicking its arrow. This will enable the configuration links for that particular integration.



## 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.

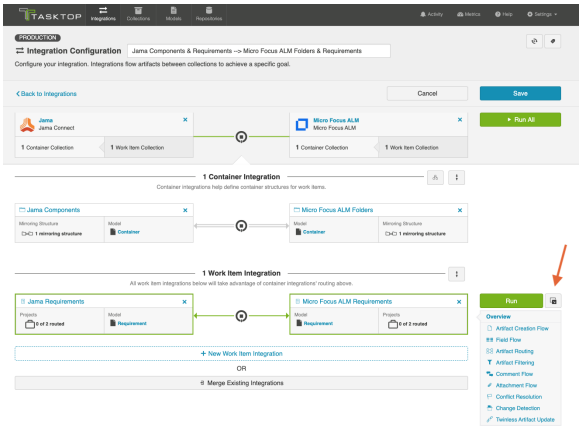




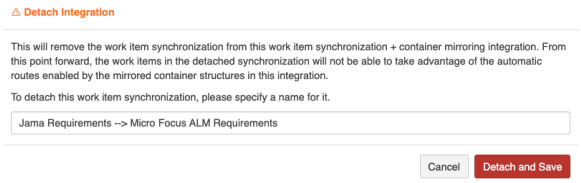

## Detaching a Work Item Integration

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).

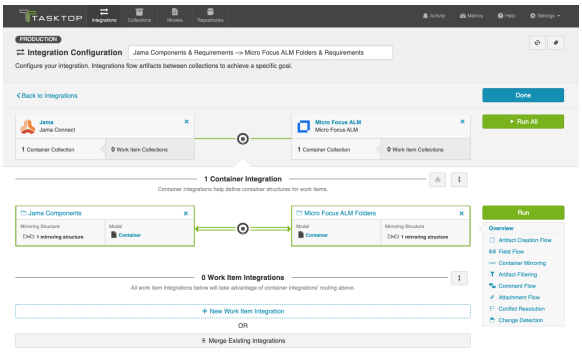
Next, click the **Detach** button.



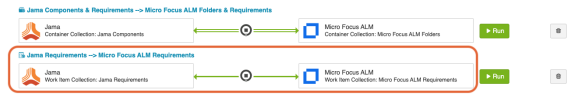

You will be prompted to name your integration:




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:

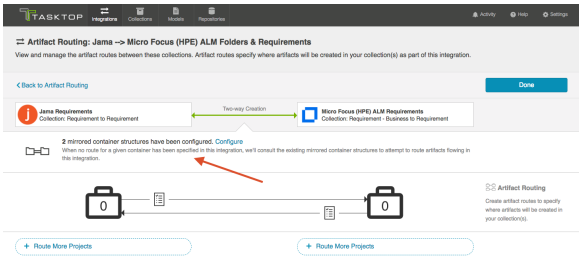



## Configuring Your Work Item Integration

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 pag e 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.



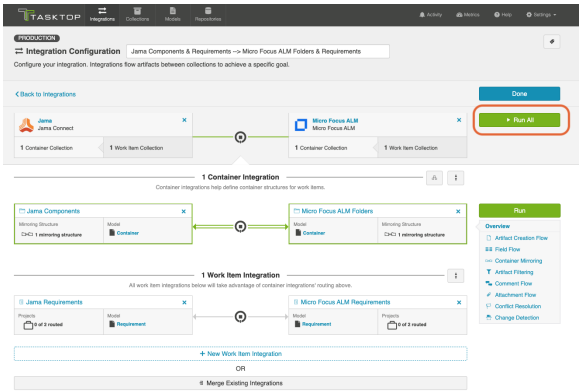

Where applicable, your work items will flow in accordance with the Container Mirroring that has been configured. In addition to the routing that is inherited based on Container Mirroring, Artifact Routing can be configured on this page to determine where work items will flow with regard to containers 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.

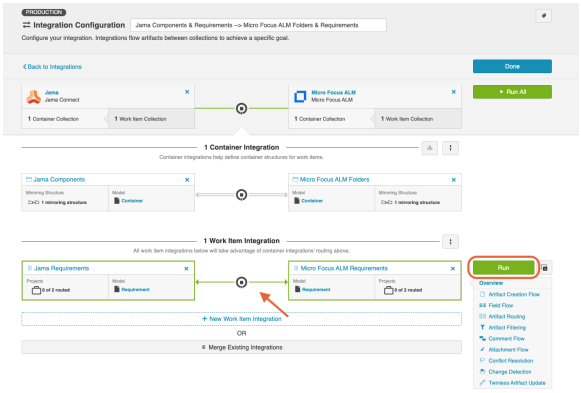
## Running your Integration

⚠️ 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.

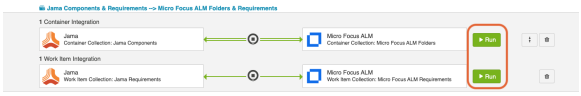
Since your Container + Work Item Synchronization technically consists of several independent, but interconnected integrations, you can select **Run All** to run all integrations at once, or choose to run integrations independently.



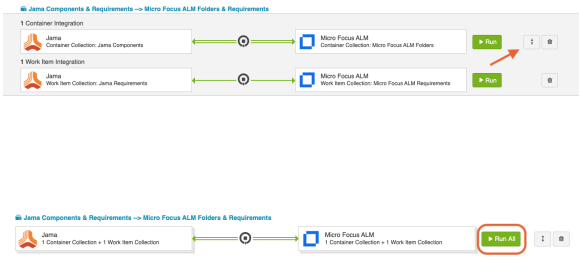
If for any reason you'd like to run an integration individually, activate that integration's configuration pane by clicking on its arrows, and then click **Run**.




You can also view and run your integration(s) from the Integration List screen. On this screen, your integration will defaulted to the expanded view, where you can run each integration individually:



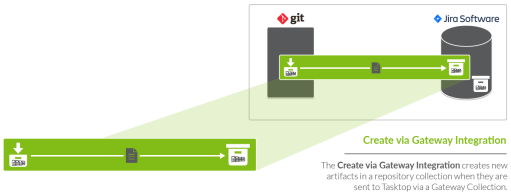

If you'd like to **Run All**, you can collapse the view and then click **Run All**:



# Create via Gateway

*The Create via Gateway Integration Template is only available in Editions that contain the Gateway add-on. See the Tasktop Editions table to determine if your edition contains this functionality.*

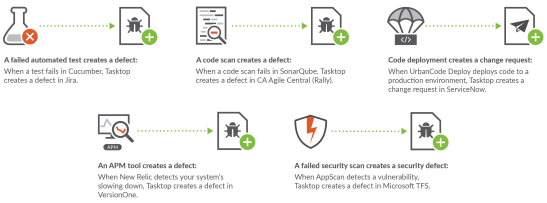
## What is a Create via Gateway Integration?




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

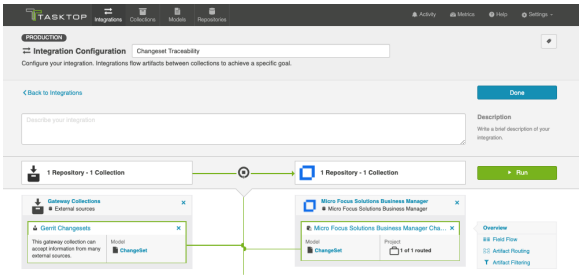
Check out the video below to learn how to configure the Create via Gateway Integration Template.

⚠️ This video assumes that you have already configured your repositories, models, and collections as outlined in the Quick Start Guide .

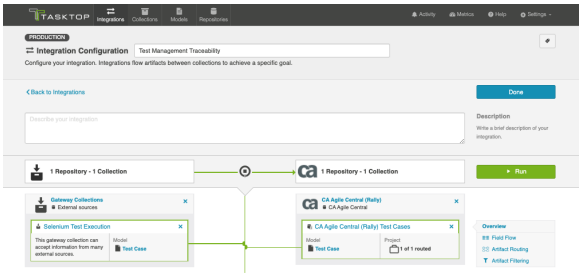
# Use Case and Business Value

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.

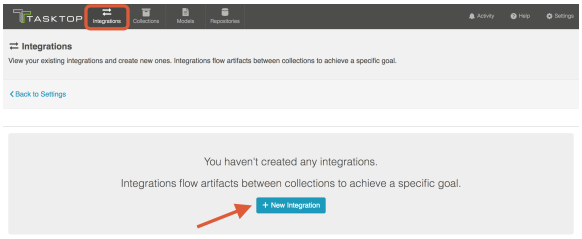



# Template Affordances

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.

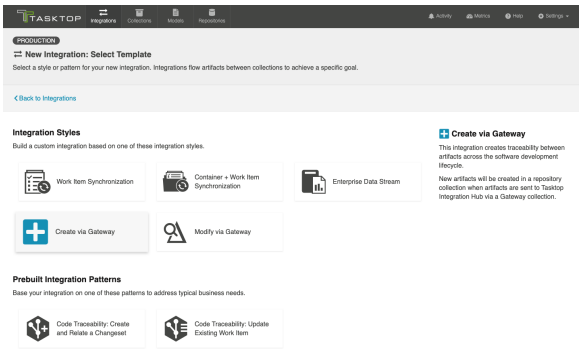

# Configuring a Create via Gateway Integration

To configure your integration, select **Integrations** at the top of the screen, then click **New Integration**.

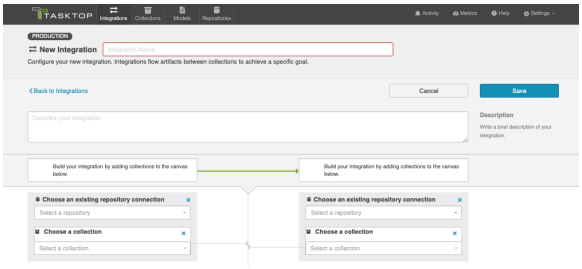



Select the **Create via Gateway** template.

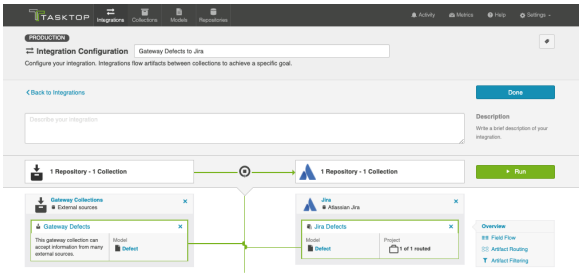
💡 Depending on the edition of Tasktop you are utilizing, you may not see all options shown below.




This will bring you to the New Integration screen.

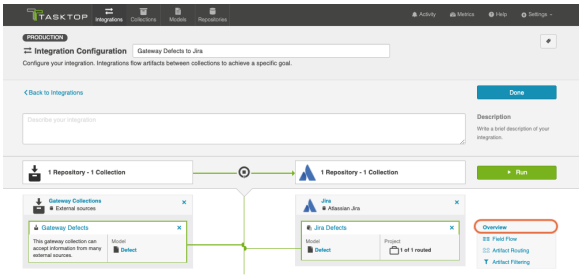


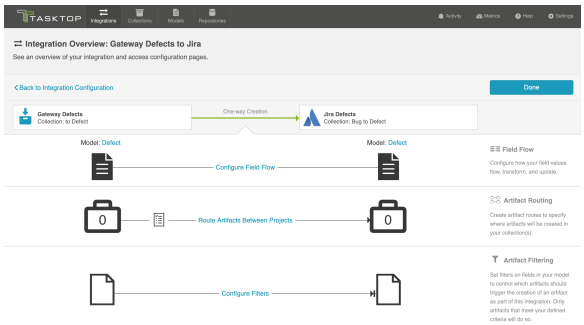
Name your integration and select your repositories and collections.



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.

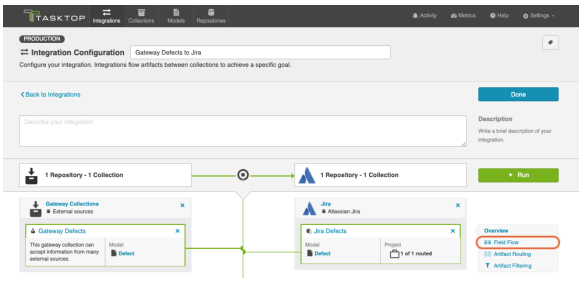




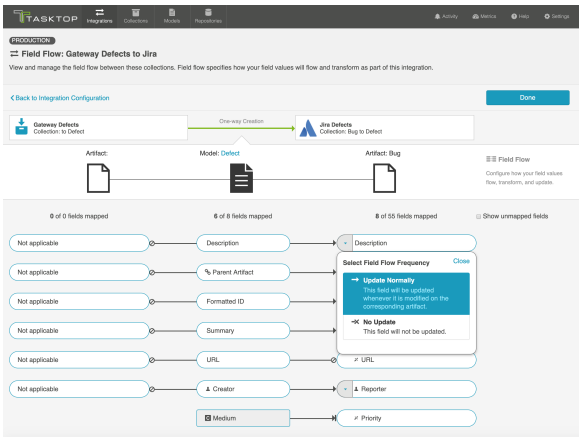
# Field Flow

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.

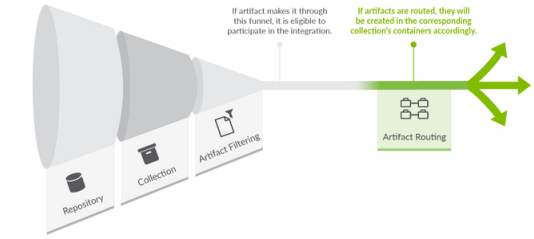
## Field Flow Icons

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 |
| --- | --- |

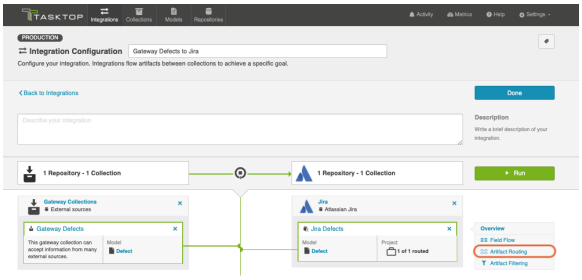
| | |
|---|---|
| C | A constant value will be sent.<br><br>Note that:<br><br>• If the icon is on the side of the collection, this means that a constant value will be sent to your model. This means that any time this collection is integrated with another collection, the *other* collection will receive this constant value for the field in question.<br>• 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 *this* collection will receive this constant value for the field in question. |
| | A state transition will be utilized. Note that:<br><br>• If the icon is on the side of the collection, this means that a state transition graph is being utilized.<br>• If the icon is on the side of the model, this means that a state transition extension is being utilized.<br><br>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. |
| | 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 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.

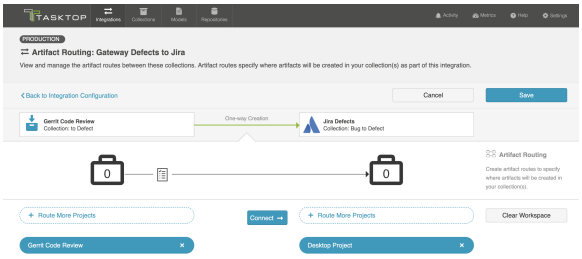



## Static Artifact Routing

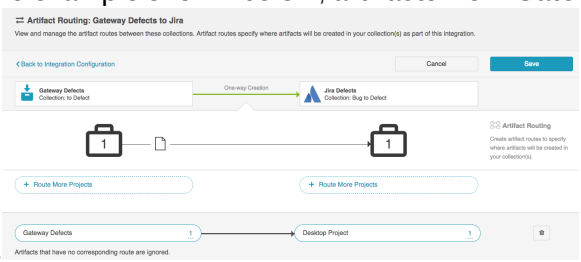
In some cases, all artifacts in a gateway collection are routed to just one project in the target collection. In these instances, you can configure what is known as 'static artifact routing' (also known as 'explicit artifact routing').

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.



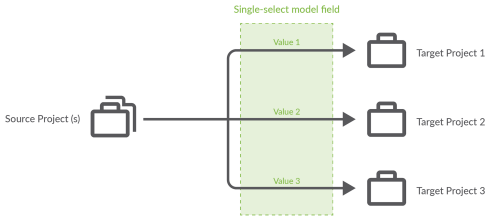

## Conditional Artifact Routing

Check out the video below to learn more about Conditional Artifact Routing:

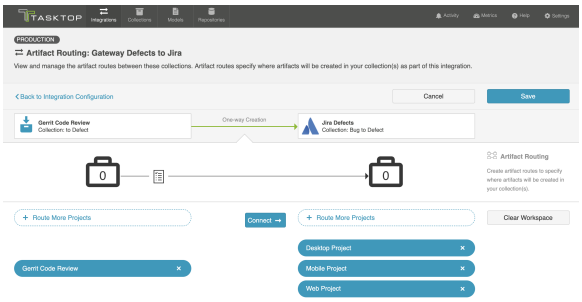
In other cases, you may wish to route your gateway artifacts to multiple projects in the target collection.  In this scenario, a field value on the artifact is used to determine which project in the target collection the artifact should route to.

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.

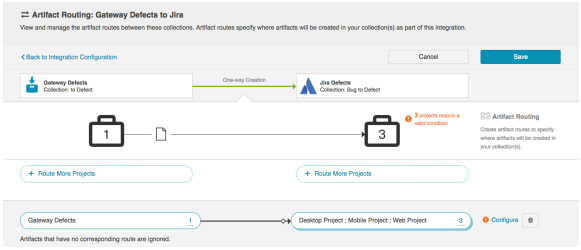
Conditional artifact routes can have one or more source projects, and always have multiple target projects.




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.



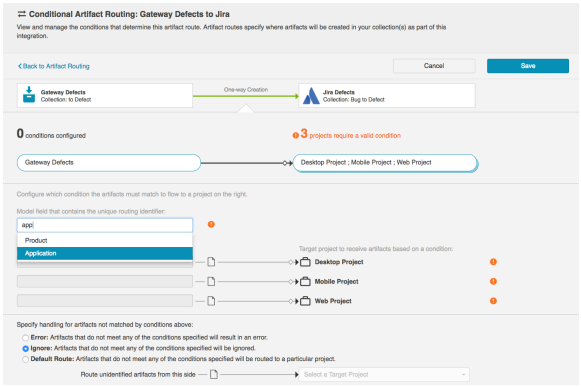

Notice that after you've created your conditional artifact routing group, you'll be prompted to set the conditions that will define that route.



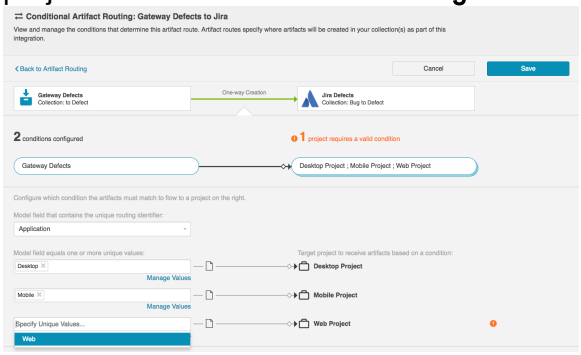

Click **Save** and then click **Configure**. You'll be brought to the Conditional Artifact Routing screen. Here you'll start by selecting the model field on the artifact that you would like to use to determine your artifact route.

💡 **Note**: Conditional Artifact Routes can only be configured based on **single-select fields** in your model.

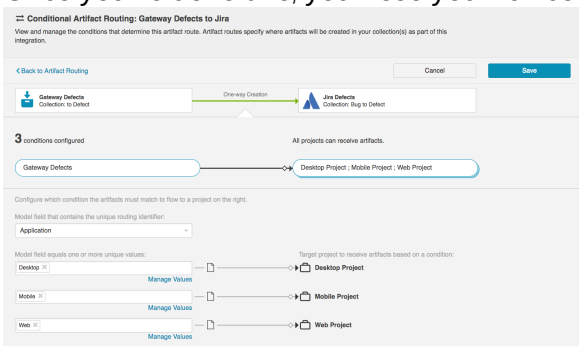
In the example below, the field **Application** contains the unique values that should determine the project an artifact will be created in in Jira.



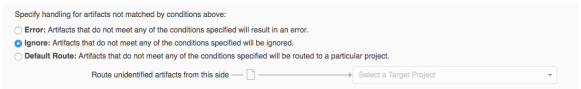

After you select the model field, you can identify one or more value to correspond to each target project. You can also use the **Manage Values** link to select from a list of values.




Once you've done this, you'll see your full conditional artifact routing group.

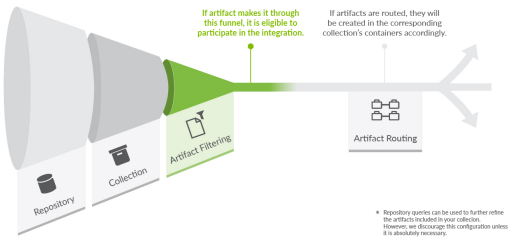



You can also specify how you'd like to handle artifacts that do not meet any of the conditions specified by selecting one of the options provided at the bottom of the screen:




# 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.

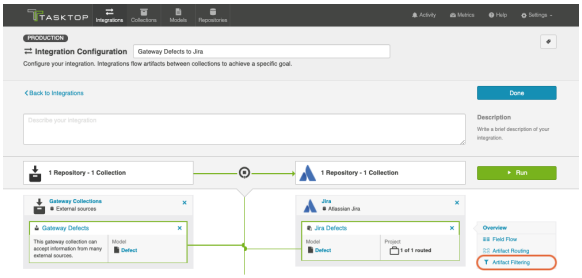



**Artifact Filtering** enables you to set filters in order to limit which artifacts are eligible to flow in an integration.

To use a field for artifact filtering, it must:

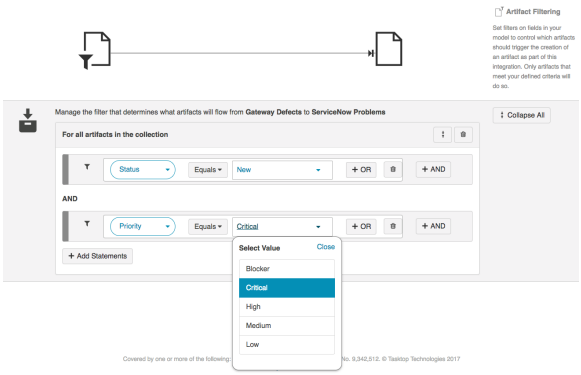
- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for 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 easier-to-read summary of your artifact filtering statements.



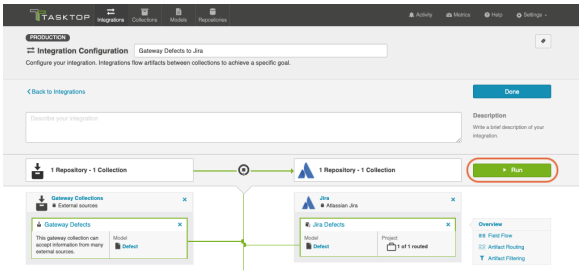

# Running your Integration

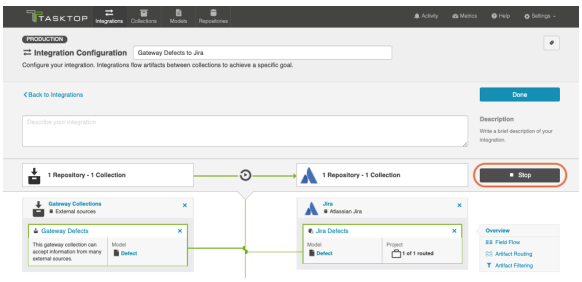
⚠️ Please be aware that integrations will trigger changes in your end repositories and that misconfiguration can cause items to be duplicated or modified in unexpected ways. Additionally, there is no 'undo' in Tasktop or the repositories. If you have any questions, please contact support.

There are two ways to start or stop your integration:

## From the Integration Configuration Screen

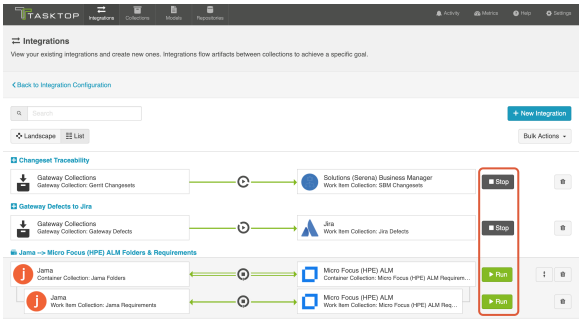
Simply click the **Run** button to run the integration, and the **Stop** button to stop the integration.

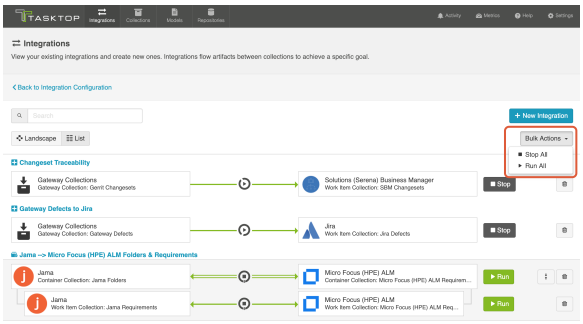



## From the Integrations List Page

Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** b utton to run or stop all integrations.

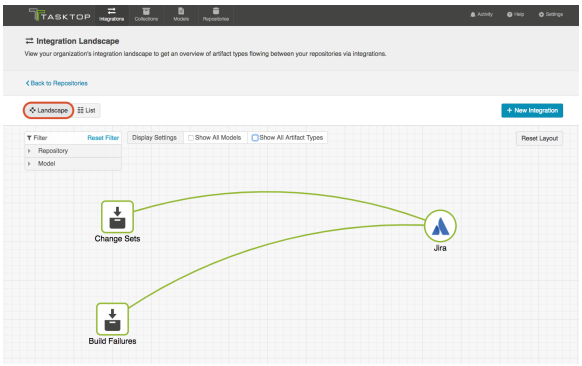


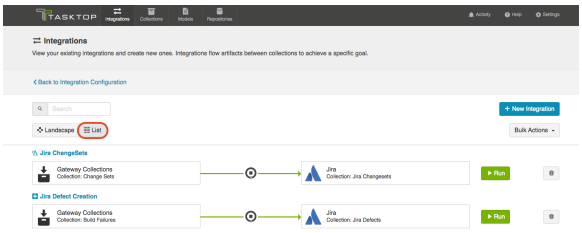


# Viewing Your Integrations

*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.



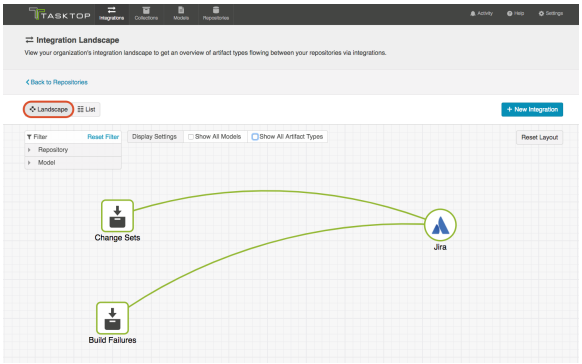
# Landscape View

*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:

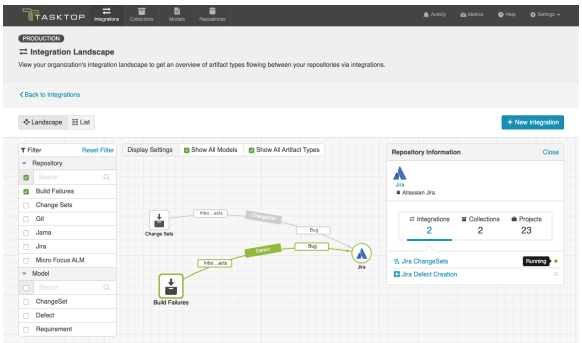



If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

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:

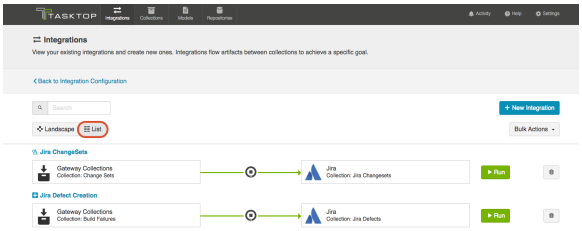



## List View

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

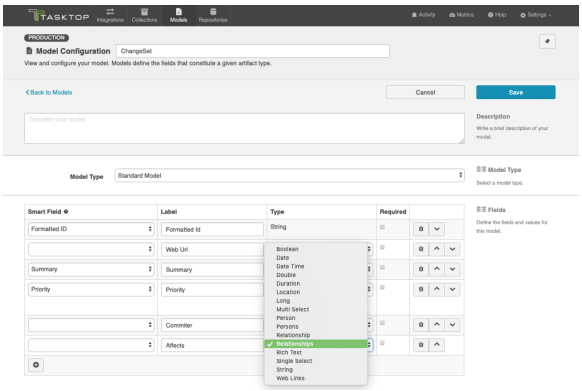
## Creating Relationships Between Newly Created Artifacts and Existing Artifacts

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.

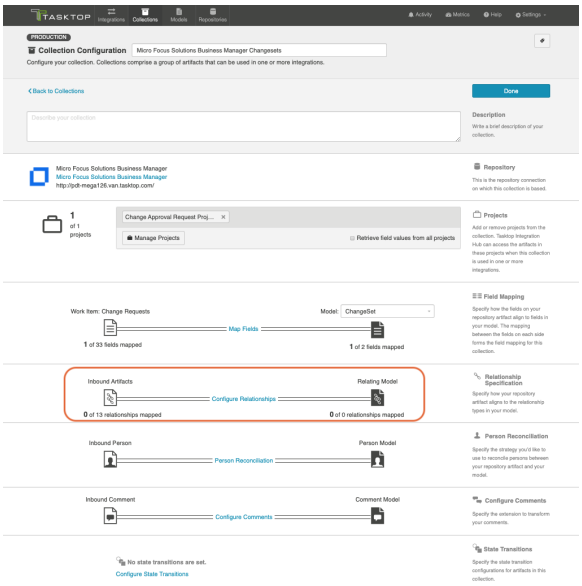
For example, the relationship field type, “Parent," should generally be singular, as most artifacts usually only have a single parent. However, if the relationship field type is called “Blocks”, it can likely be plural, as one artifact can block many artifacts.

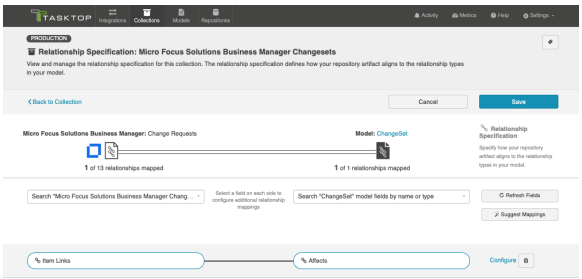
In the use case example described at the top of this page, I want the relationship to be “Affects” because any incoming changeset can affect many stories. So I’ll configure a *relationships* field.



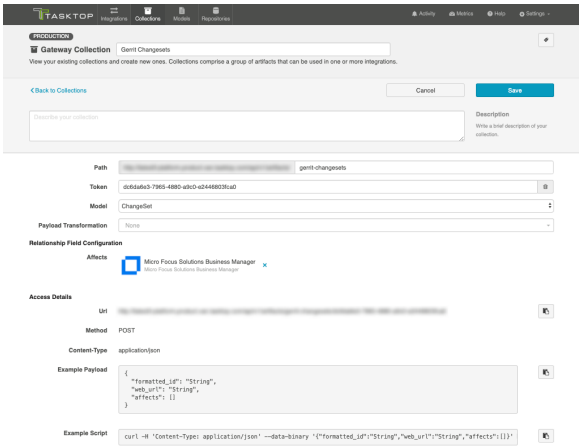

**At the Repository Collection level**: When creating your repository collection, you will need to map a field in your repository to the relationship(s) field in your model. So, in the same example, if you want

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.



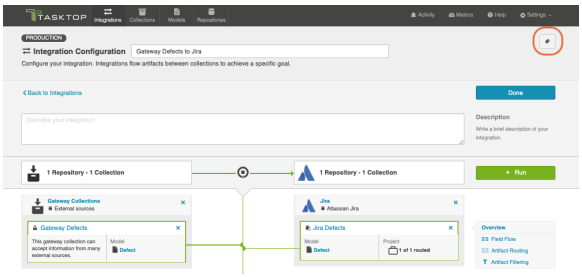



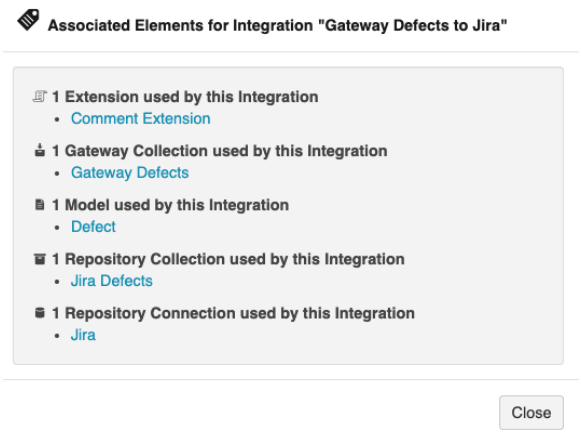

**At the Gateway Collection Level:** When creating your gateway collection, you will see that for each model field that is of relationship(s) type, you must specify the target repository that contains the related artifact(s). Once this is selected, the information needed for Tasktop to successfully locate 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.

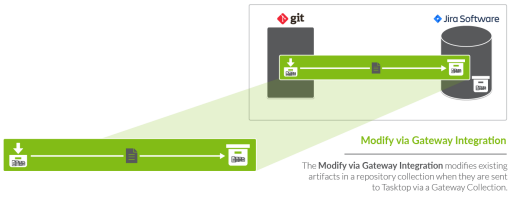




# Modify via Gateway

*The Modify via Gateway Integration template is only available in Editions that contain the Gateway add-on. See the 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

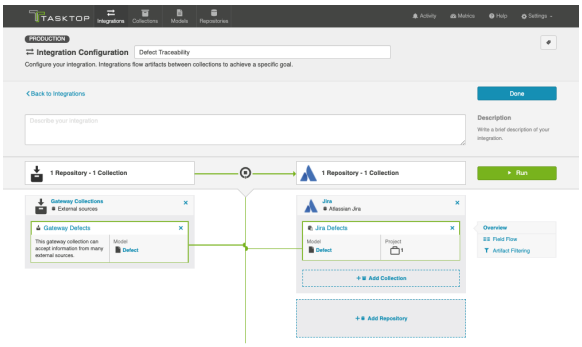
Check out the video below to learn how to configure the Modify via Gateway Integration template.

⚠️ 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.

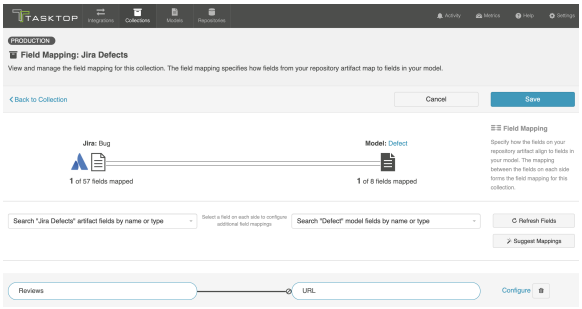

# Configuring a Modify via Gateway Integration

## Configuring your Repository Collection

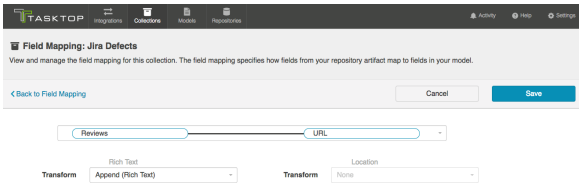
Before you begin configuring the integration itself, there are some steps that must be taken at the repository collection level:

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.

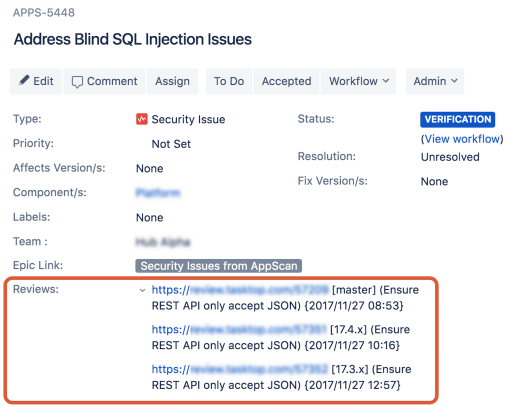
This means that the Jira collection-to-model mapping is configured as such:




And here are how the transformations are configured between these fields:

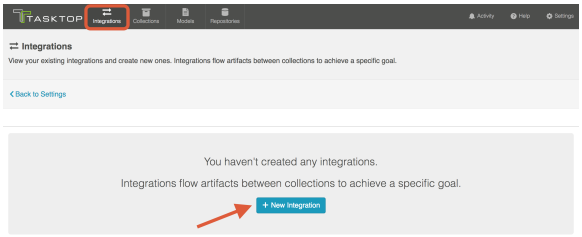



The **Append** transform means that new values will be added to the field value, rather than overwriting it, leaving the Jira artifact itself looking like this:



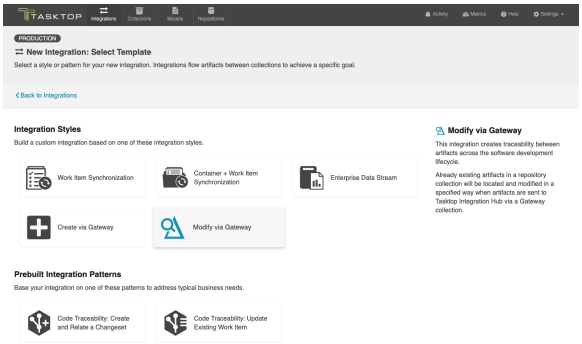

# Configuring Your Integration

To configure your integration, select **Integrations** at the top of the screen, then click **New Integration**.

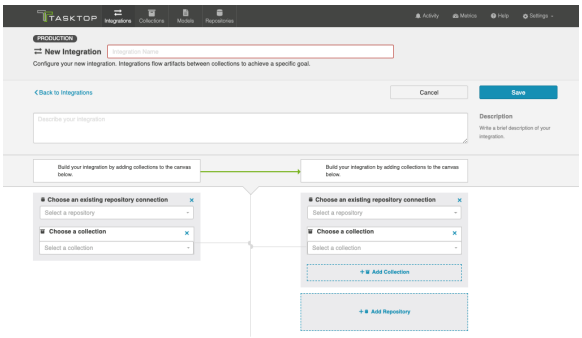



Select the **Modify via Gateway** template.

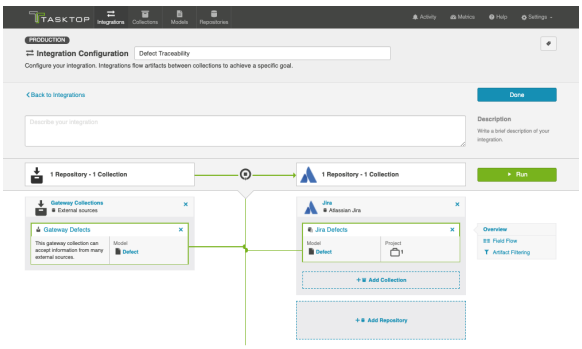
Depending on the edition of Tasktop you are utilizing, you may not have all options shown here.



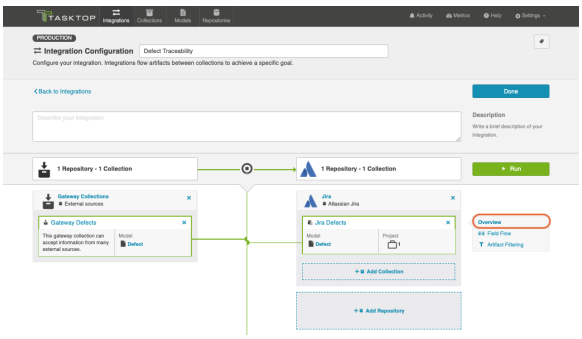

This will bring you to the New Integration screen:

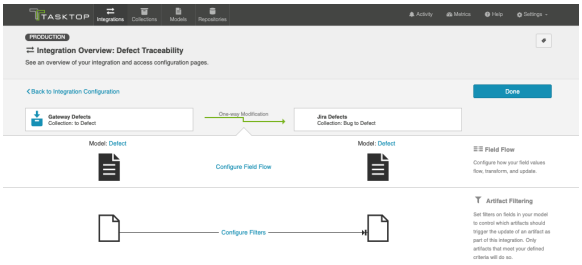



Name your integration and select your repositories and collections:




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):

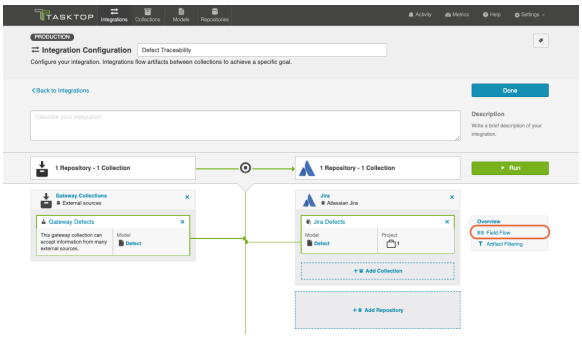




# Field Flow

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.



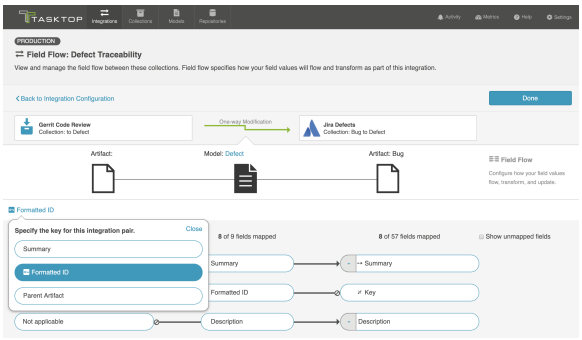
## Specifying Your Key

The first thing you will need to do when you get to the Field Flow screen is to specify your key.

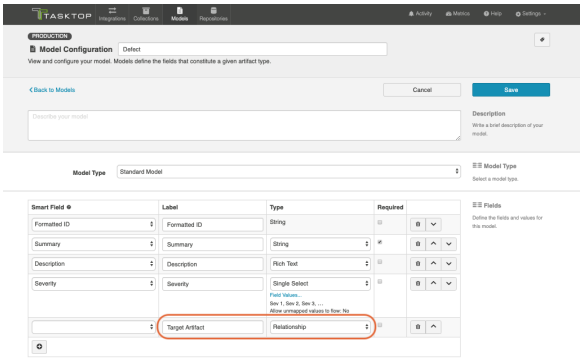
Specifying a key will enable Tasktop to find the existing artifact in your repository collection that is to be modified by the incoming gateway payload(s). The key can be a string or relationship field from the model.

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.

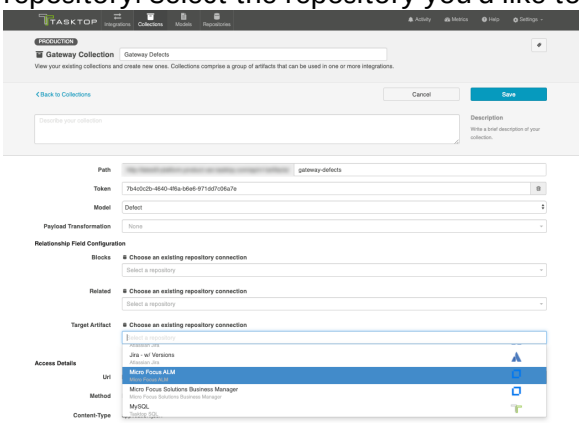
If the key is a relationship field, then the artifact it references in the gateway payload will be used as the target artifact.



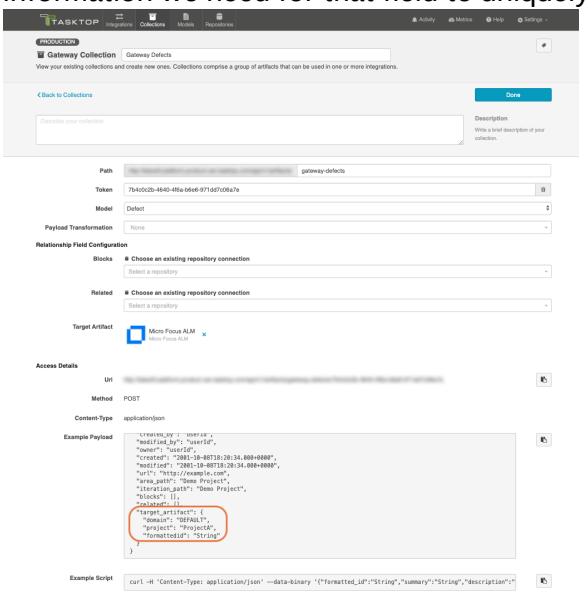

💡 **Note**: Some repositories require extra information in order to uniquely identify a single artifact across multiple projects. One prime example is ALM. To ensure that enough information is sent in via your gateway collection to allow Tasktop to find the specific artifact you would like to modify, please take these steps:



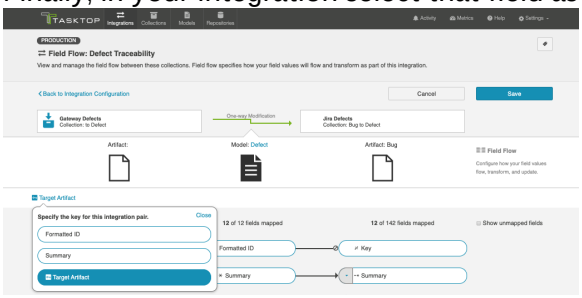

1. Add a field in your model of type relationship.
2. In your gateway collection, notice that for the new field you are prompted to pick a target repository. Select the repository you'd like to target in this gateway integration.



3. 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.



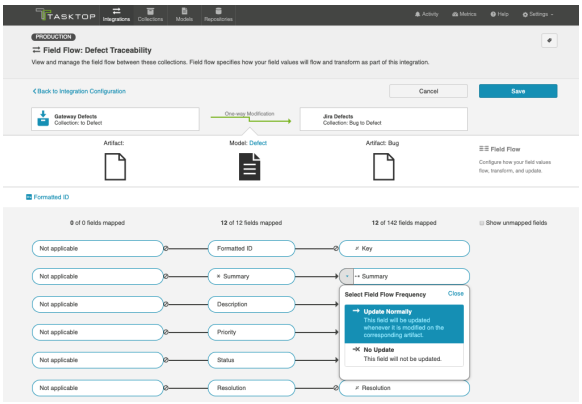

4. Finally, in your integration select that field as your key on the Field Flow screen.




## 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 |
|---|---|
| C | A constant value will be sent.<br><br>Note that:<br><br>• 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 *other* collection will receive this constant value for the field in question.<br>• 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 *this* collection will receive this constant value for the field in question. |
| | A state transition will be utilized. Note that:<br><br>• If the icon is on the side of the collection, this means that a state transition graph is being utilized.<br>• If the icon is on the side of the model, this means that a state transition extension is being utilized.<br><br>💡 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. |
| | 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. |

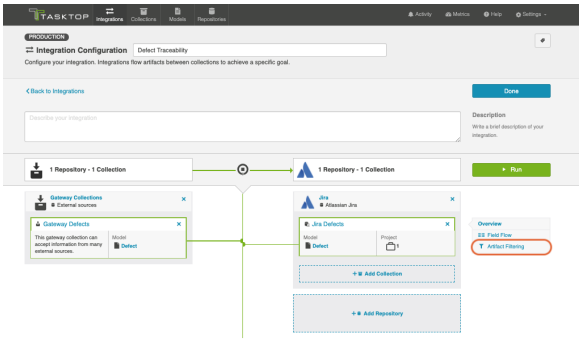
# Artifact Filtering

**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:

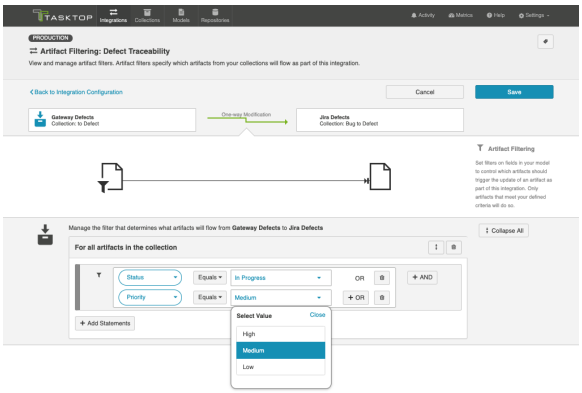
- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for 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.



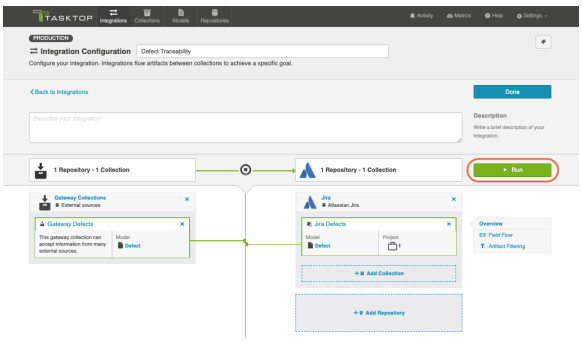

## Running your Integration

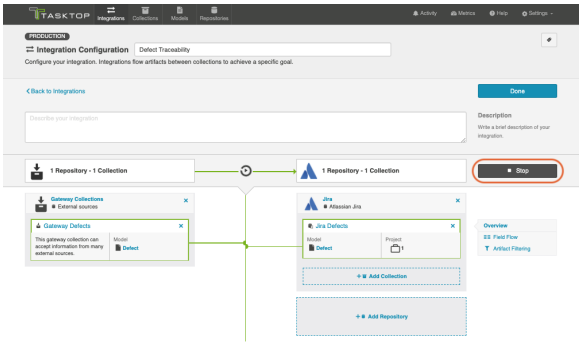
⚠️ Please be aware that integrations will trigger changes in your end repositories and that misconfiguration can cause items to be duplicated or modified in unexpected ways. Additionally, there is no 'undo' in Tasktop or the repositories. If you have any questions, please contact support.

There are two ways to start or stop your integration:

# From the Integration Configuration Screen

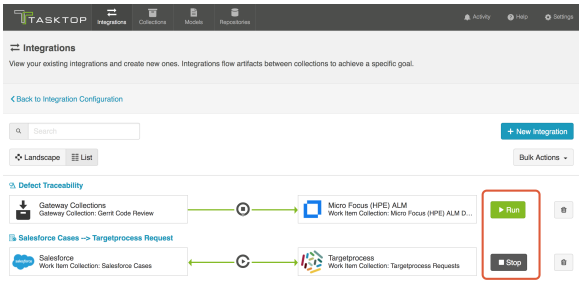
Simply click the **Run** button to run the integration, and the **Stop** button to stop the integration.

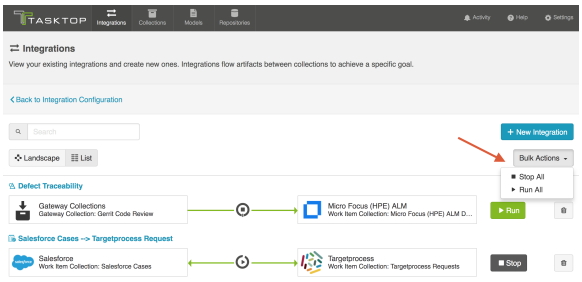


## From the Integrations List Screen

Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** b utton to run or stop all integrations.

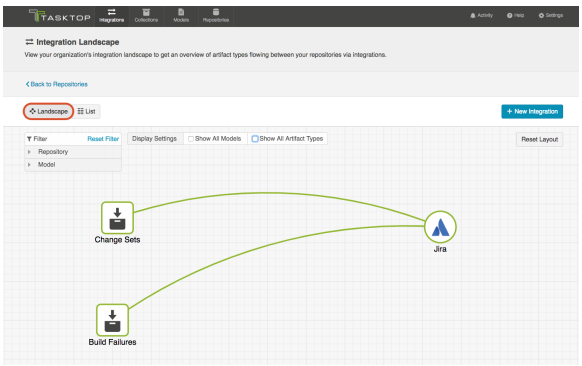


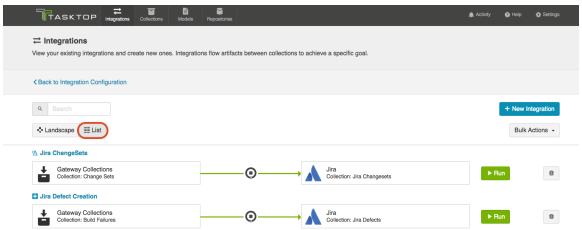




# Viewing Your Integrations

*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.



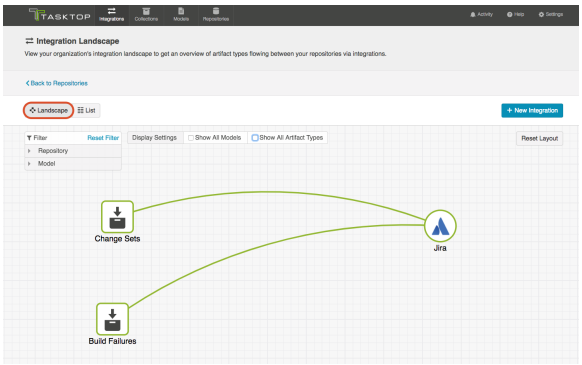
# Landscape View

*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:

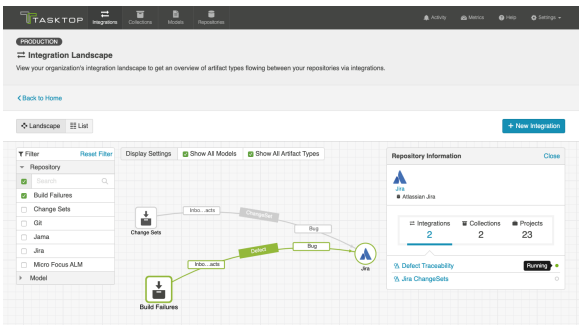



If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

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:

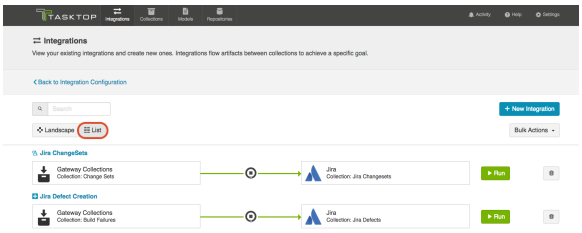



# List View

If you'd like, you can toggle to List view, which will show you a list of all integrations you have created.

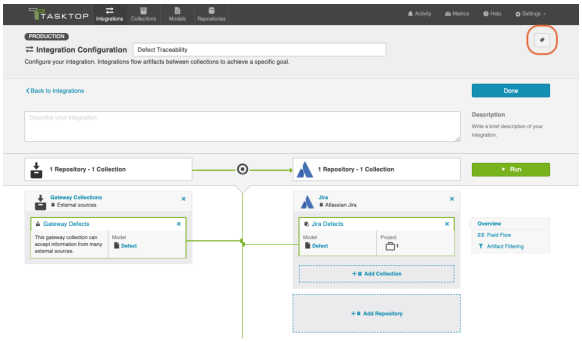
You can use this view to:

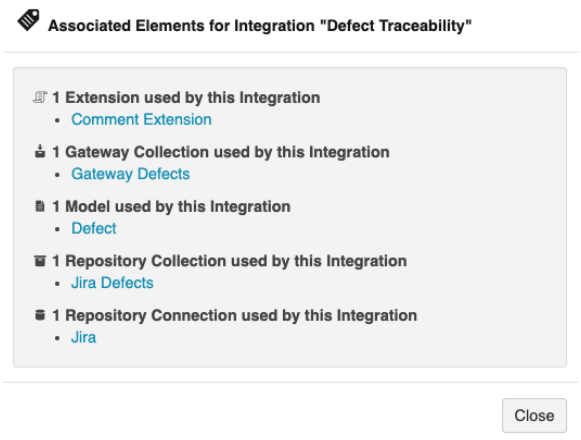
- Start an Integration
- Stop an Integration
- Delete an Integration
- Click into an Integration and modify its configuration




## 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.

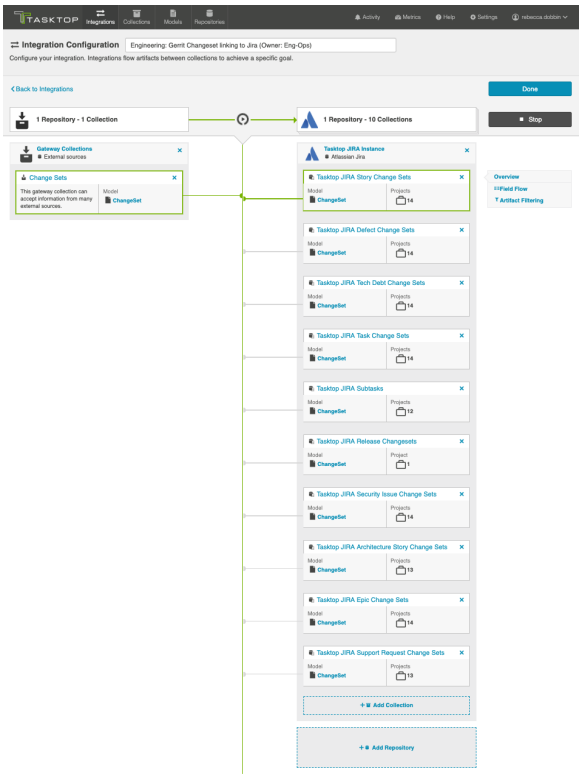


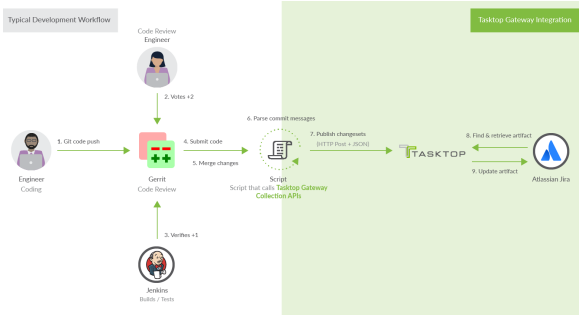

# Example Use Case

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:



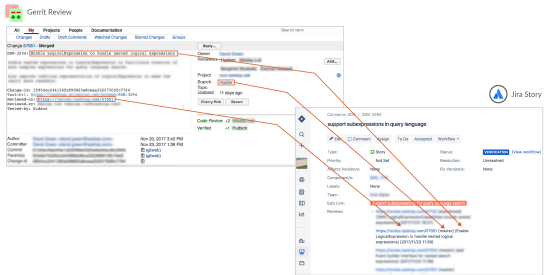

This is an example of the script that we use to automate the changesets being sent to Tasktop:

**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
```

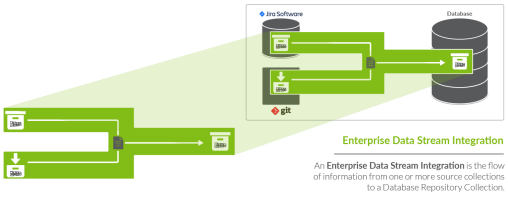
This image more clearly highlights how these changesets are reflected on the Jira artifacts:



# Enterprise Data Stream

*The Enterprise Data Stream Template is only available in Editions that contain the Enterprise Data Stream add-on. See the 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

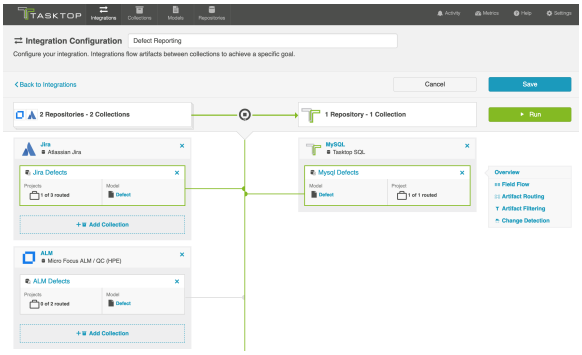
Check out the video below to learn how to configure an Enterprise Data Stream Integration.

⚠️ 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

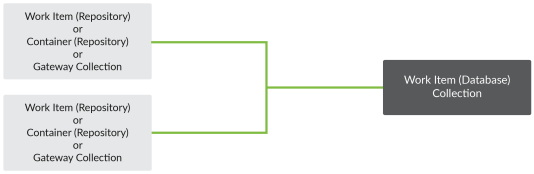
This integration simplifies enterprise reporting by unlocking software lifecycle data from its application tool silos and providing a rich data repository for near real-time analytics. Records will be created in a single database when artifacts from one or more collections are created or changed.

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 t hat 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 `artifact_id` is an opaque value; assumptions should not be made about its structure or content. It should be noted that the `artifact_id` 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."

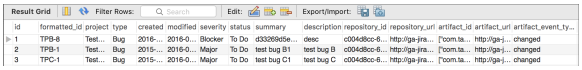
#### Changed

Changed indicates that either an existing artifact was changed or that a new artifact was added to your collection.

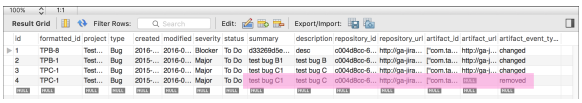
#### Removed

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:



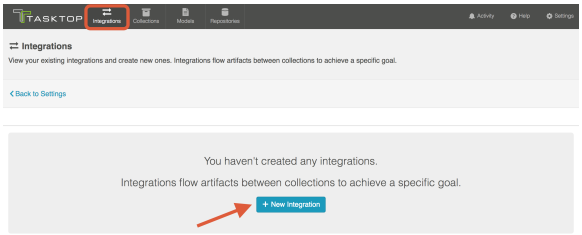

Project C is then removed from the source collection. At next full scan (one of the change detection intervals configured on the General (Settings) screen), you'll see an event to denote that any artifacts in that collection have been removed:




⚠️ 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.

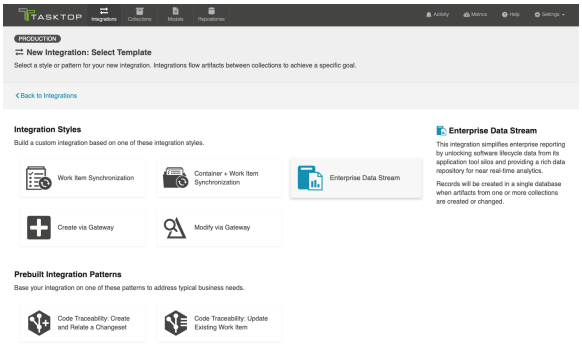
# Configuring an Enterprise Data Stream Integration

To configure your integration, select **Integrations** at the top of the screen, then click **New Integration**.

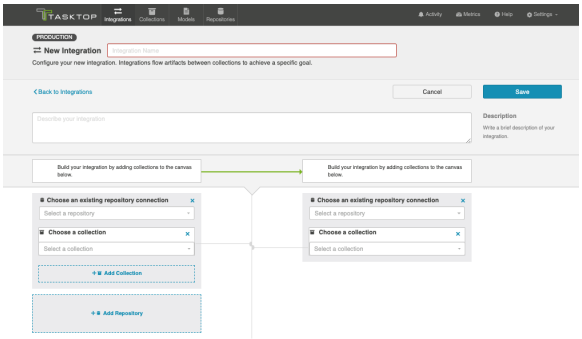



Select the **Enterprise Data Stream** template.

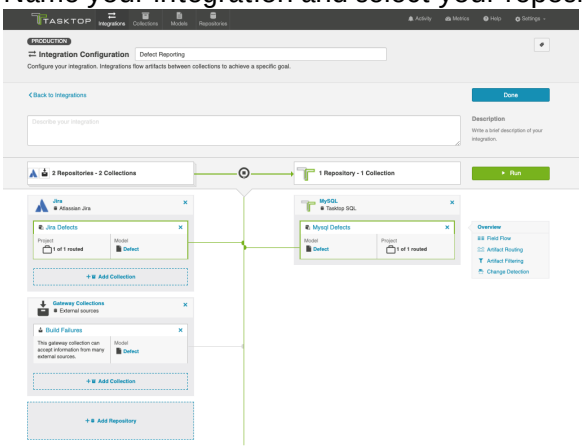
💡 Depending on the edition of Tasktop you are utilizing, you may not see all options shown below.




This will bring you to the New Integration screen:

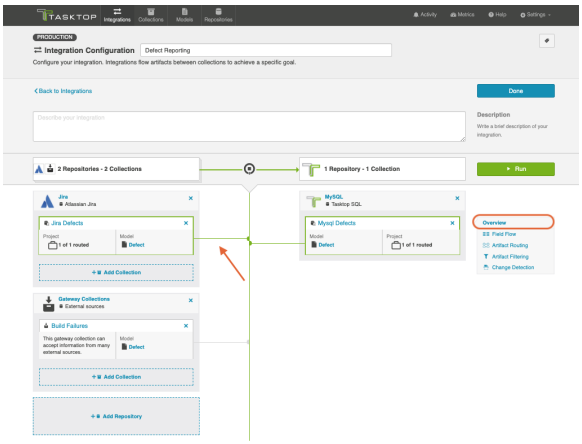


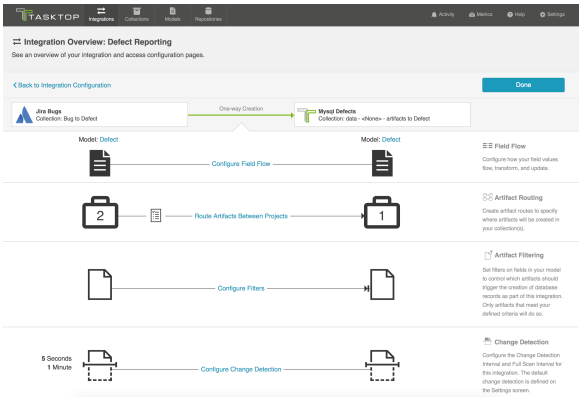

Name your integration and select your repositories and collections.



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**.

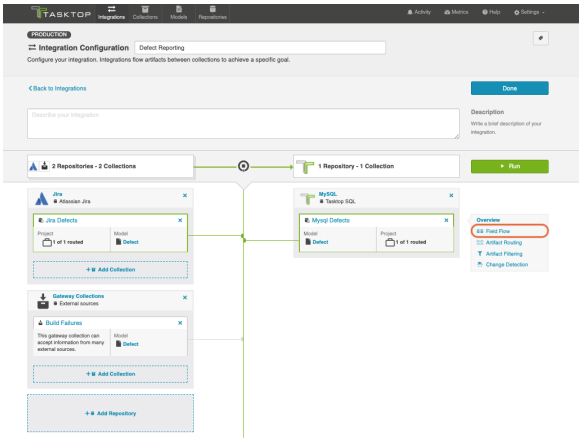


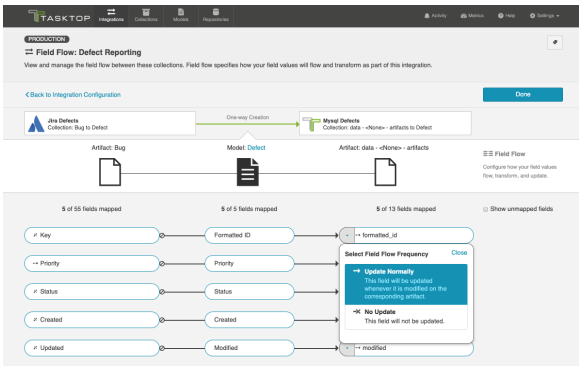

# Field Flow

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.

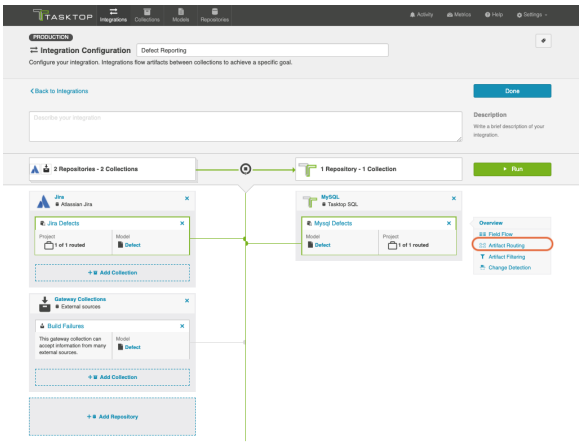
⚠ **Note**: The field flow settings behave a bit differently for Constant Values. This is because constant values exist as part of your Tasktop configuration, and not on the artifact itself. Therefore, changes in constant values are not detected in the same way that updates made on the actual artifact 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

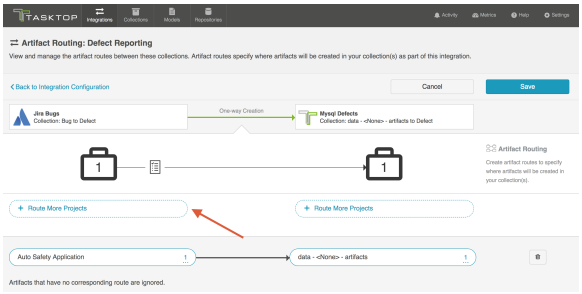
For an Enterprise Data Stream Integration, Artifact Routing is used to specify which projects (or other containers) you would like to participate in your integration.  For example, your Jira Bugs collection

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.

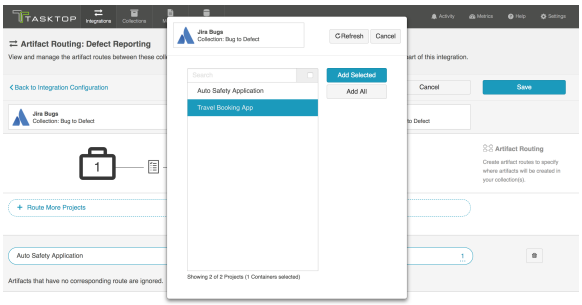
To configure Artifact Routing, select the relevant repositories and then click **Artifact Routing**.



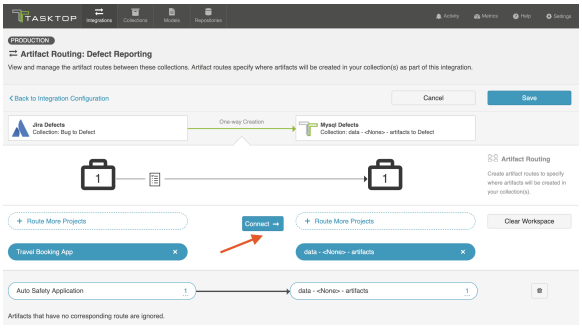

This will bring you to the Artifact Routing screen. You can click **Route More Projects** to add additional projects to your route.



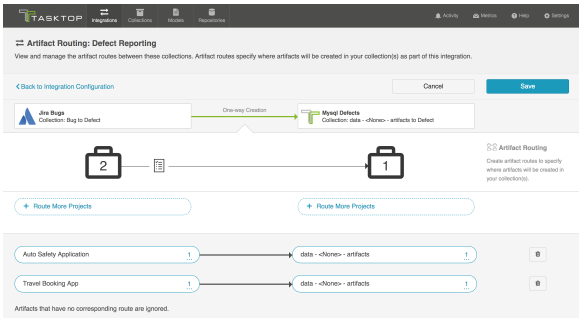

Select the projects you would like to participate in the integration and click **Add Selected**.




Click **Connect**.




You will see your artifact route on the pane below. Click **Save** and **Done**.




# 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

💡 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:

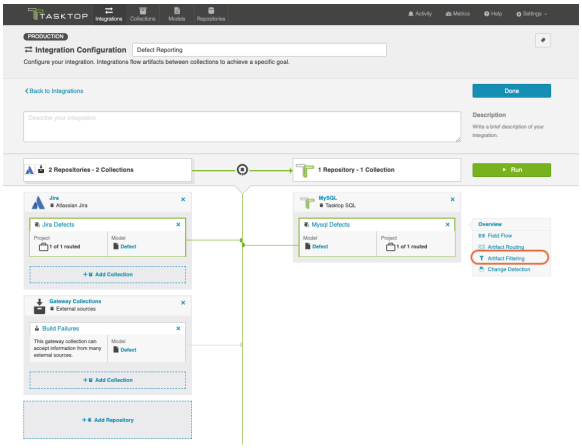
Though setting filters is meant to limit which artifacts flow in an integration, the impacts of setting filters on an Enterprise Data Stream Template are somewhat unique. Because it would not be ideal to have records in your database output that represent artifacts that have been filtered in an integration, 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.



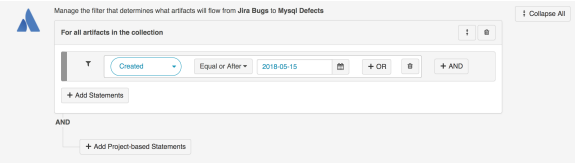
This will lead you to the Artifact Filtering Configuration screen, where you can configure your artifact filtering statement(s).

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

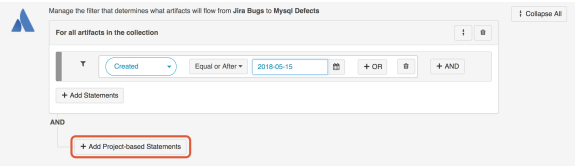
To apply a filter to all artifacts in the collection, simply click **+ Add Statements**.


Use the drop-down menus to configure your filter fields and values:

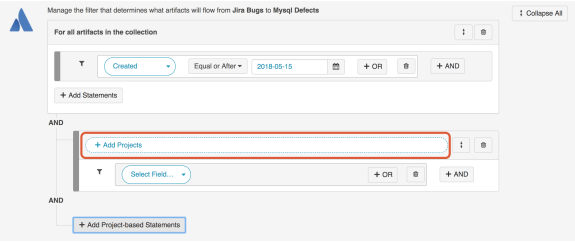



## Apply Filter to Artifacts in Certain Projects

To apply a filter to artifacts within a specific project, click + **Add Project-based Statements**.

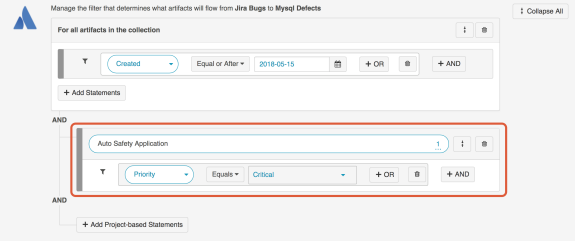



Click **+ Add Projects** to select your project.



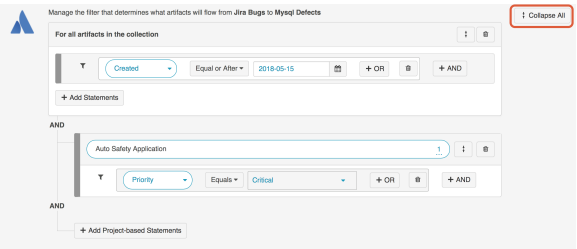

Select the project(s) you'd like your filter to apply to.

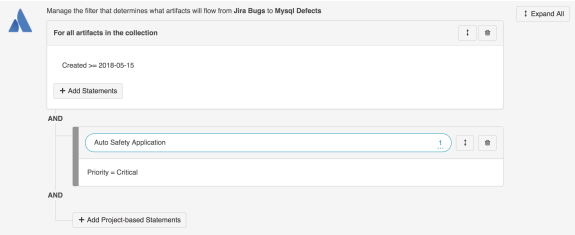
Then click **Select Field...** to begin configuring your filtering statement.



### Viewing Artifact Filter Statements

You can click the **Collapse All** button to view an easier-to-read version of your artifact filtering statements.



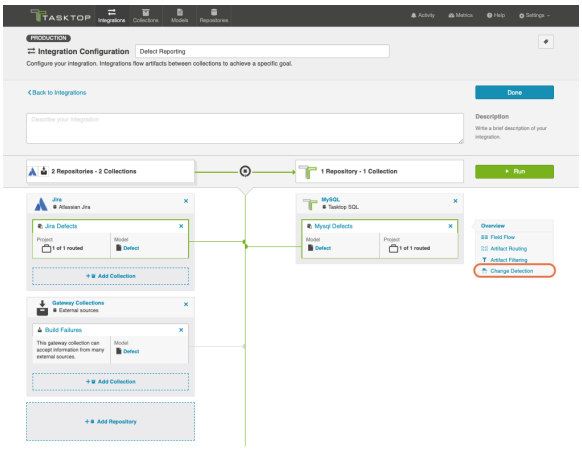

# Change Detection

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.

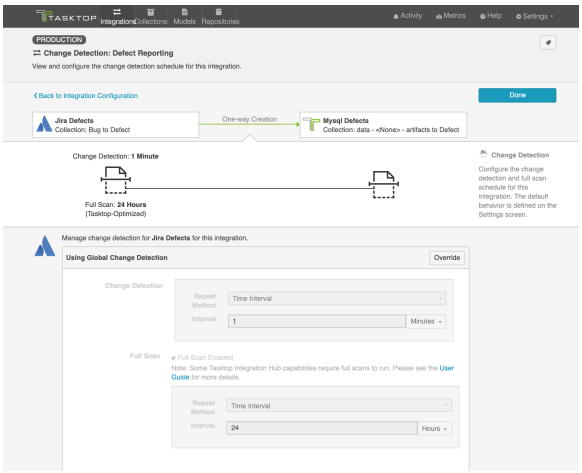
The **Change Detection Interval** is the time between polling requests to detect *only changed artifacts*. This defaults to 1 minute on the General (Settings) screen, but can be customized as desired.

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.

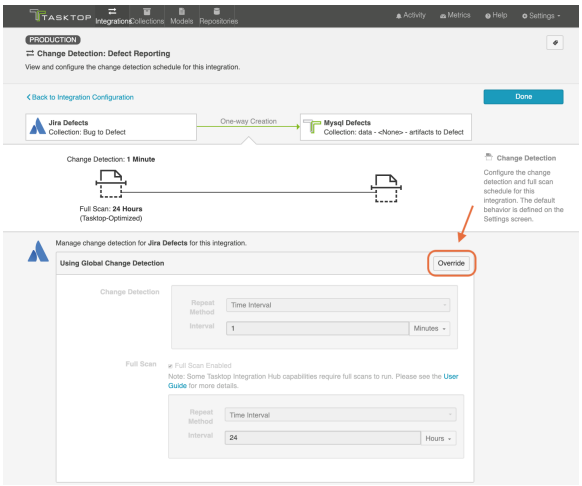
To configure integration-specific change detection, select the relevant repository, then click the **Change Detection** link.



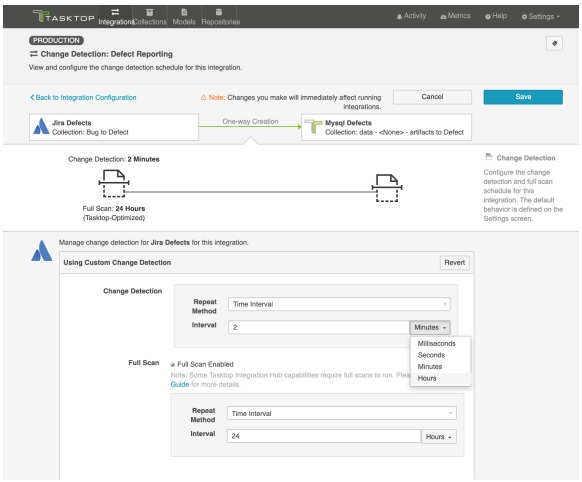
This will bring you to the Change Detection screen, where you can view the current change detection and full scan intervals configured for the selected collection in this integration. These will default to the global intervals configured on the General (Settings) screen.



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.



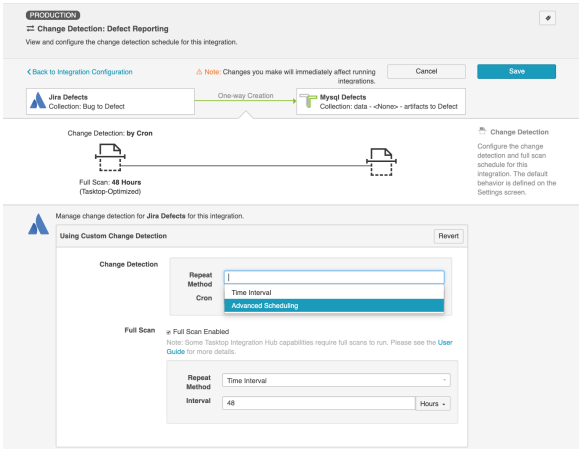
Once you click **Override**, you will be able to configure custom change detection and full scan intervals for the collection within the context of this integration.




In addition to customizing change detection and full scan intervals, you can also schedule change detection or full scan using cron expression.

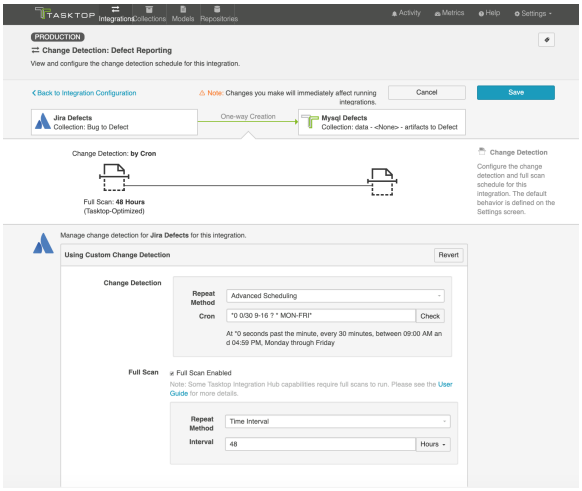
Using cron expression, you can configure complex schedules by running change detection or full scan during certain hours of the day, certain days of the week, or specific days of the month. Learn more in our FAQ here.

To utilize cron expression for your full scan or change detection, select **Advanced Scheduling** in the Repeat Method field.

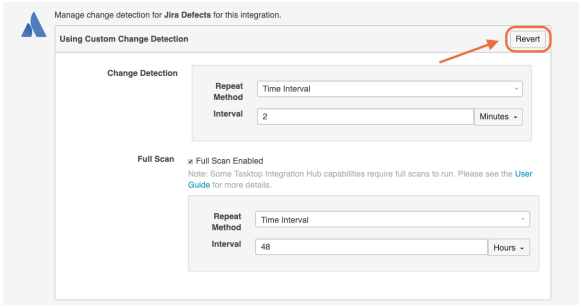



You can then enter the cron expression for your desired scheduling. For example, if you would like to run a full scan every 30 minutes from 9am to 5pm from Monday through Friday, it would be written as follows: *0 0/30 9-16 ? * MON-FRI*

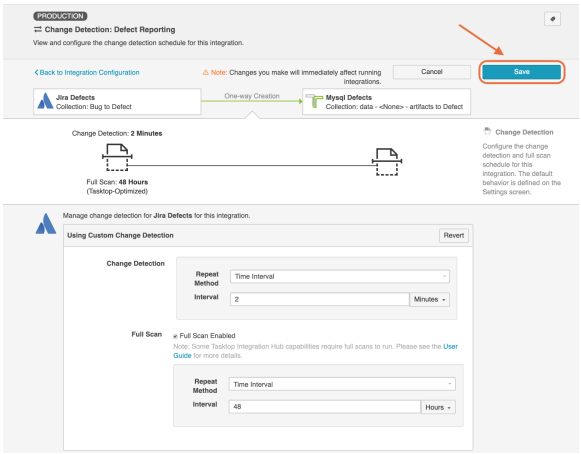
To ensure that your cron expression is valid, click the **Check** button. If valid, a readable form of the cron expression will be displayed. If the cron expression is invalid, an error message will appear.




If you'd like to restore the global change detection settings, simply click the **Revert** button to remove the custom settings.

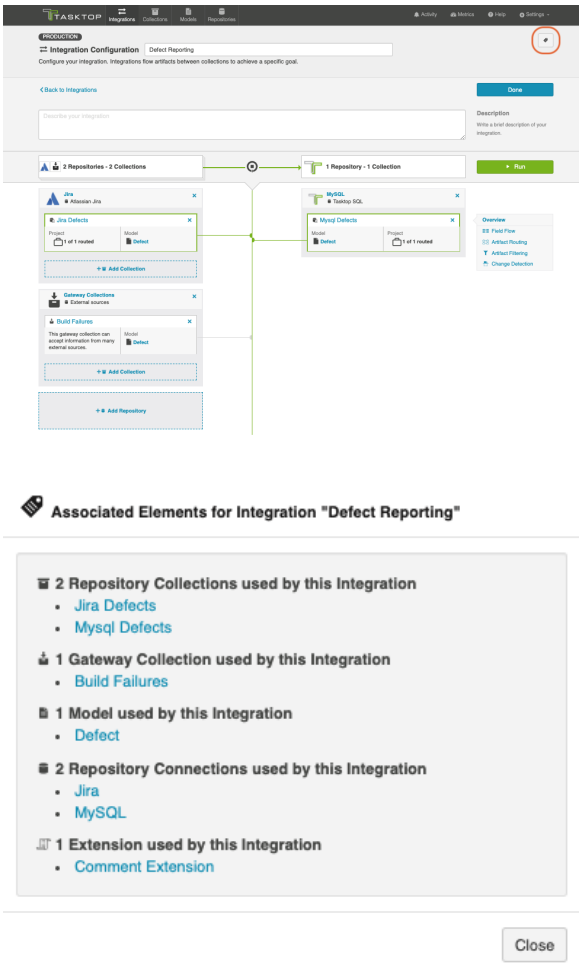



Once you've updated the change detection settings as desired, click **Save** and **Done** to save your changes.




# 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.



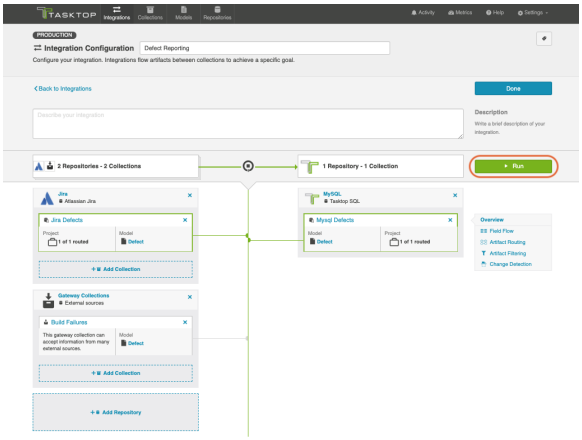

# Running your Integration

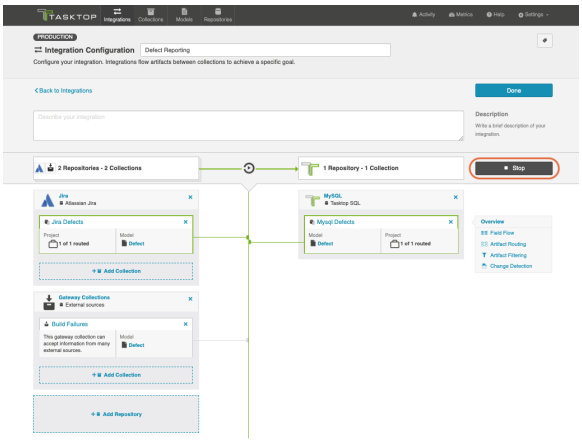
⚠️ Please be aware that integrations will trigger changes in your database and that misconfiguration can cause items to be duplicated or updated in unexpected ways. Additionally, there is no 'undo' in Tasktop or the database. If you have any questions, please contact support.

There are two ways to start or stop your integration:

## From the Integration Configuration Screen

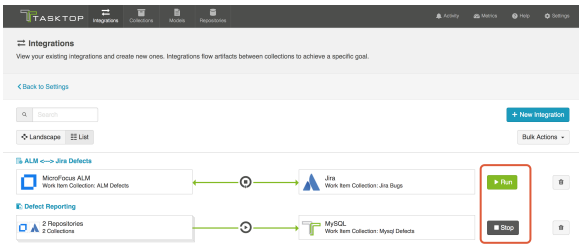
Simply click the **Run** to run the integration, and the **Stop** button to stop the integration.

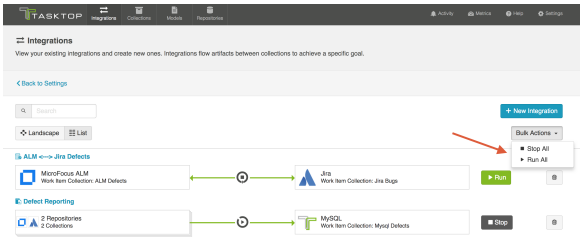




## From the Integrations List Screen

Click **Run** or **Stop** next to each integration you would like to update. You can also use the **Bulk Actions** b utton to run or stop all integrations.

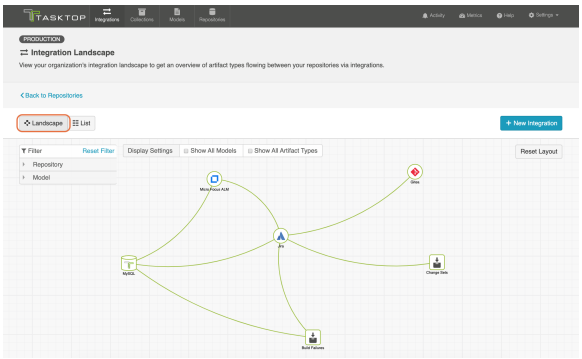


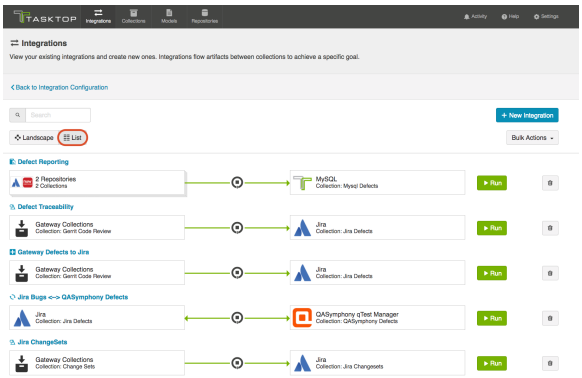


## Viewing Your Integrations

*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.



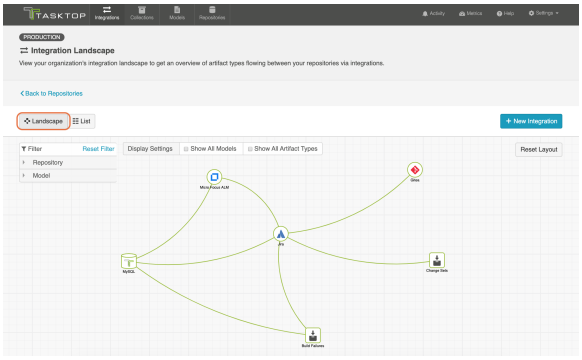

## Landscape View

*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:

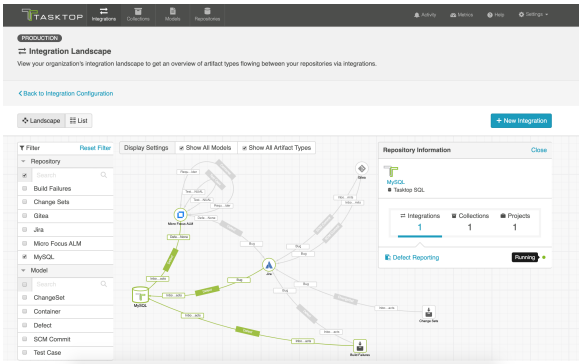



If you'd like to see additional information, you can utilize the filters, or click on a repository node to modify which information is shown.

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:

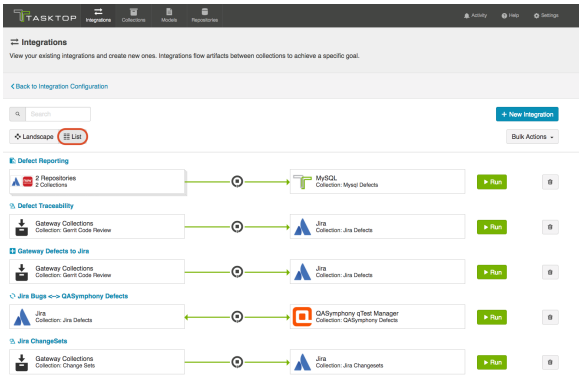



## List View

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.

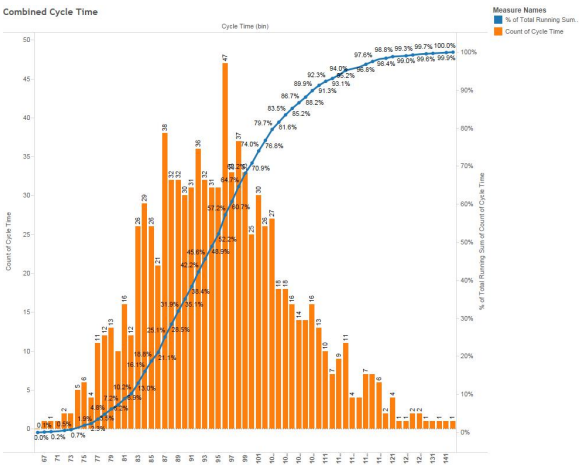
#### Artifact Cycle Time Model

| 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

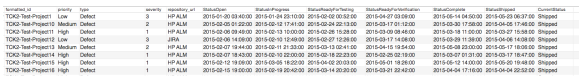



### SQL

```
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.



| formatted_id | priority | type | severity | repository_url | StatusOpen | StatusInProgress | StatusReadyForTesting | StatusReadyForVerification | StatusComplete | StatusShipped | CurrentStatus |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TCK2-Test-Project1 | Low | Defect | 3 | HP ALM | 2015-01-20 03:45:00 | 2015-01-24 23:10:00 | 2015-02-02 00:52:00 | 2015-04-27 03:09:00 | 2015-05-14 04:50:00 | 2015-05-25 06:37:00 | Shipped |
| TCK2-Test-Project10 | Medium | Defect | 2 | HP ALM | 2015-02-05 01:22:00 | 2015-02-12 17:41:00 | 2015-02-24 22:13:00 | 2015-03-17 01:12:00 | 2015-03-30 17:58:00 | 2015-04-05 17:48:00 | Shipped |
| TCK2-Test-Project11 | High | Defect | 1 | HP ALM | 2015-02-06 09:49:00 | 2015-02-13 10:00:00 | 2015-02-26 15:26:00 | 2015-03-09 08:46:00 | 2015-03-11 00:00:00 | 2015-03-27 15:58:00 | Shipped |
| TCK2-Test-Project12 | Low | Defect | 3 | JIRA | 2015-02-06 14:09:00 | 2015-02-10 12:49:00 | 2015-02-27 12:26:00 | 2015-03-17 14:08:00 | 2015-03-29 11:39:00 | 2015-04-06 14:08:00 | Shipped |
| TCK2-Test-Project13 | Medium | Defect | 2 | HP ALM | 2015-02-07 19:44:00 | 2015-02-11 21:33:00 | 2015-03-13 02:41:00 | 2015-04-15 19:54:00 | 2015-05-08 23:00:00 | 2015-05-17 18:08:00 | Shipped |
| TCK2-Test-Project14 | High | Defect | 1 | HP ALM | 2015-02-07 18:43:00 | 2015-02-10 22:00:00 | 2015-03-18 22:23:00 | 2015-05-25 02:19:00 | 2015-05-27 01:31:00 | 2015-03-17 19:47:00 | Shipped |
| TCK2-Test-Project15 | High | Defect | 1 | HP ALM | 2015-02-12 19:09:00 | 2015-03-05 18:22:00 | 2015-04-02 20:03:00 | 2015-05-01 18:26:00 | 2015-05-12 14:00:30 | 2015-05-20 19:48:00 | Shipped |
| TCK2-Test-Project16 | High | Defect | 1 | HP ALM | 2015-02-15 19:00:00 | 2015-02-19 20:42:00 | 2015-03-14 20:20:00 | 2015-03-21 22:42:00 | 2015-04-04 17:16:00 | 2015-04-04 20:52:00 | 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.

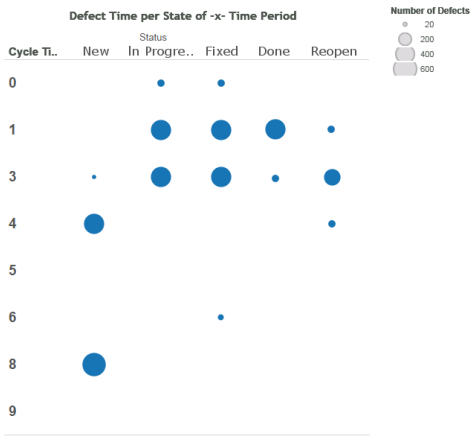
We provide a basic defect model packaged with our product:

## 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

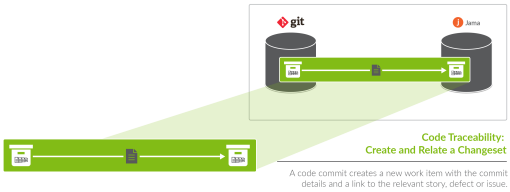
```
SELECT status, COUNT(artifact_id), cycleTime FROM (
  SELECT A.ARTIFACT_ID AS artifact_id, A.STATUS AS status, SUM(  TIMESTAMPDIFF(SQL_TSI_DAY,A.MODIFIED,B.
```

```
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

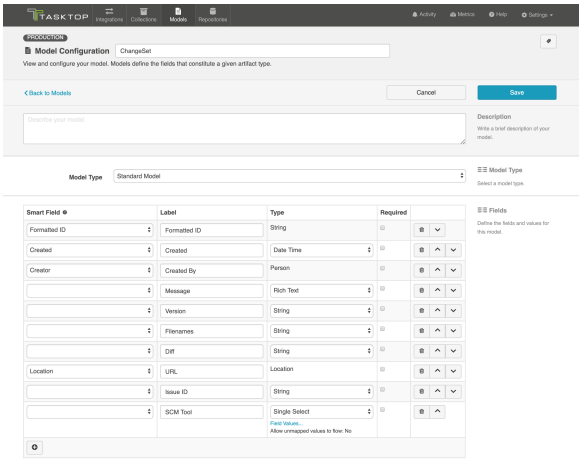
Please review the following pages to learn how to configure your repository:

- 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.

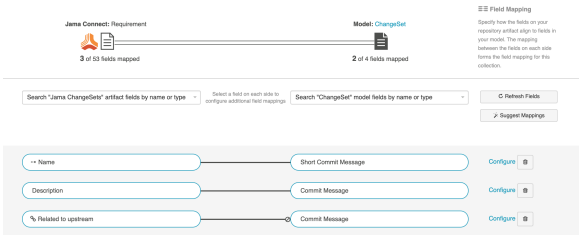
- To configure your source collection (i.e. your Git collection): Outbound Only Collection
- To configure your target collection (i.e. your Jama or Jira collection): Work Item Collection (Repository)
    - Please also review additional notes below

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<br><br>(String or Rich text) | Summary / Name |
| Commit Message<br><br>(the entire commit message) | Commit Message (String or Rich text)<br><br>*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* | Relationship of choice (i.e. 'relates to')<br><br>*see details below* |
| Commit Message<br><br>(the entire commit message) | Commit Message (String or Rich text) | Description |



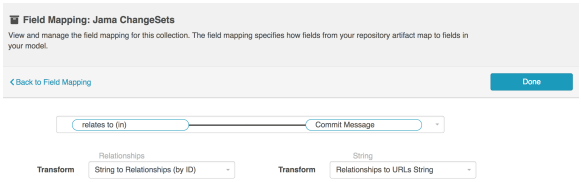

## Configuring Relationships for the Target Collection

⚠️ 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.

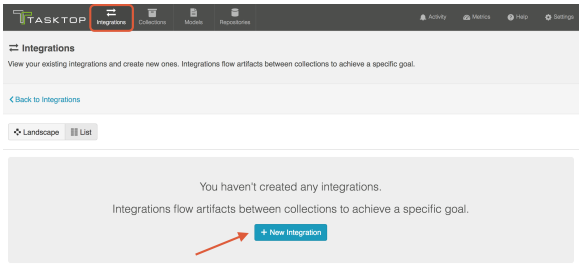
If a *relationship* field is mapped to the commit message, Tasktop will relate the newly created changeset to the first artifact ID it finds in the commit message (if there are multiple IDs). If a *relationships* field is mapped to the commit message, Tasktop will relate the newly created changeset to *all* artifact IDs it finds in the commit message.




# Configuring a Code Traceability: Create and Relate a Changeset Integration

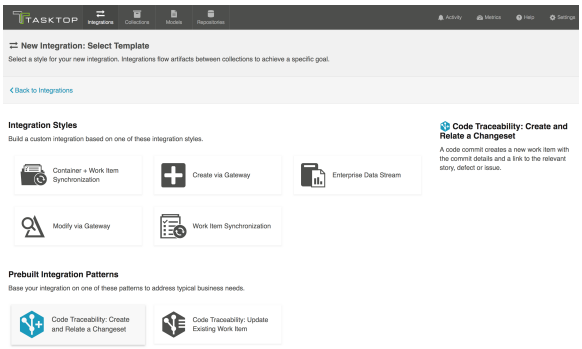
Now that you have all of your base components (i.e., repositories, models, and collections) set up, you can configure an integration to synchronize the artifacts in your collections.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

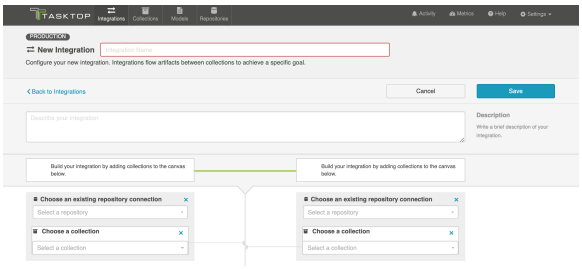



Select the Code Traceability: Create and Relate a Changeset template.

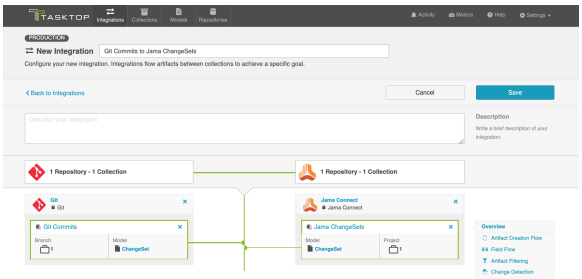
Depending on the edition of Tasktop you are utilizing, you may not have all options available.



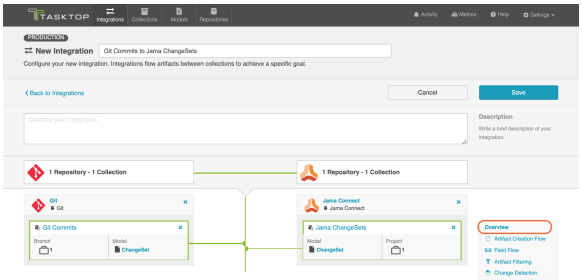

This will bring you to the New Integration screen.

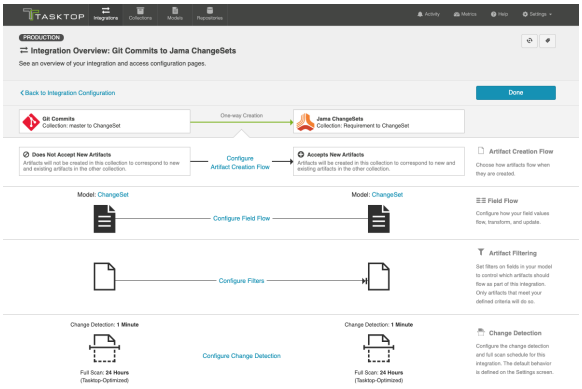



Name your integration and select your repositories and collections.



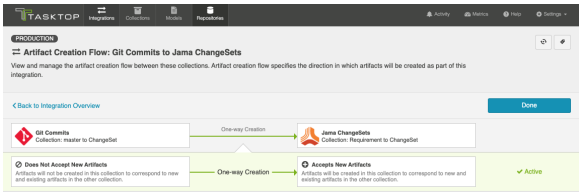

You can click the **Overview** link on the right side of the New Integration screen to get to the main display screen (shown in the second screenshot).





# Artifact Creation Flow

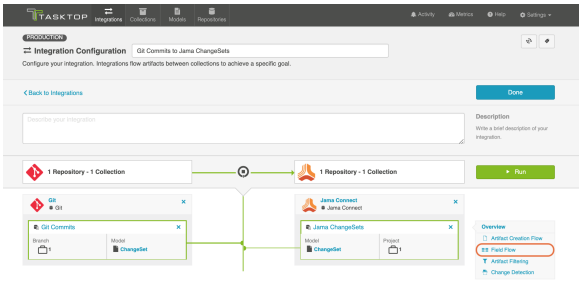
Artifacts will flow in one direction only: from the Git repository to the Work item repository. This cannot modified.



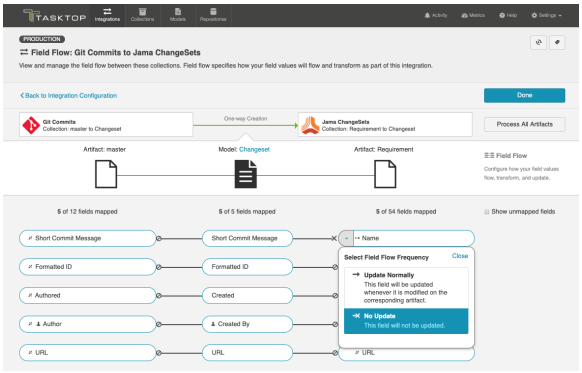
# Field Flow

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 |
| --- | --- |
| C | A constant value will be sent.<br><br>Note that:<br><br>• 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 *other* collection will receive this constant value for the field in question.<br>• 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 *this* collection will receive this constant value for the field in question. |
| | |

| | |
|---|---|
|  | A state transition will be utilized. Note that:<br>• If the icon is on the side of the collection, this means that a state transition graph is being utilized.<br>• If the icon is on the side of the model, this means that a state transition extension is being utilized.<br><br>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. |
|  | 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. |

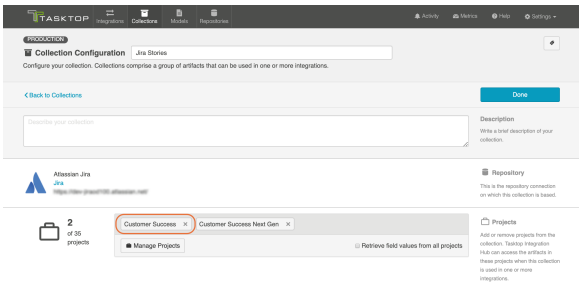
## Artifact Routing

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.



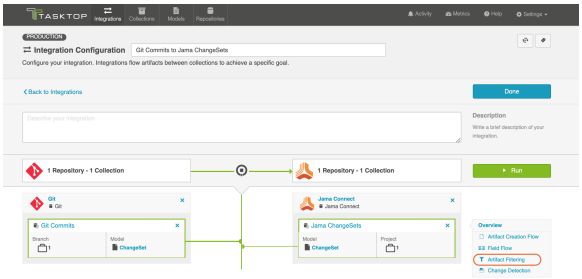

# Artifact Filtering

**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:

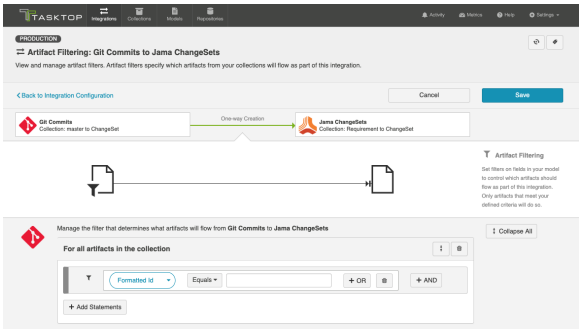
- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for 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.



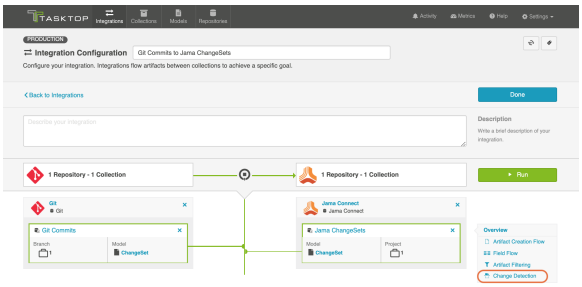
# Change Detection

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.

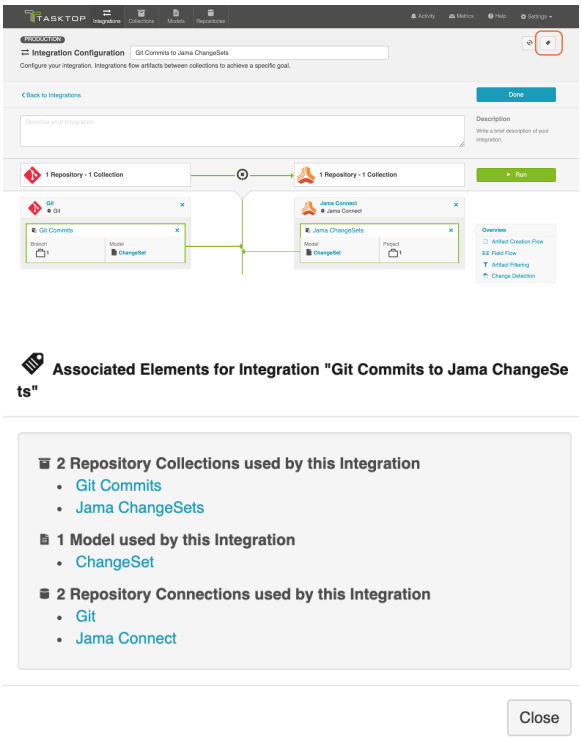
The **Full Scan Interval** is the time between polling requests to detect changed artifacts, in which *all* artifa cts of a collection are scanned. Not all changes to an artifact will register as a change. Some repositories do not mark items as changed when (for example) a relationship is added or an 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.




## Running Your Integration

To run your integration, please see details here: Running Your Integration(s)

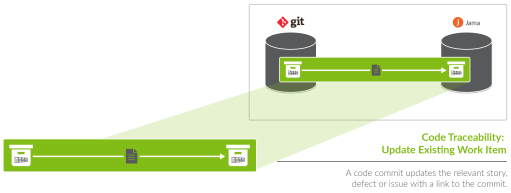
## Viewing Your Integration

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

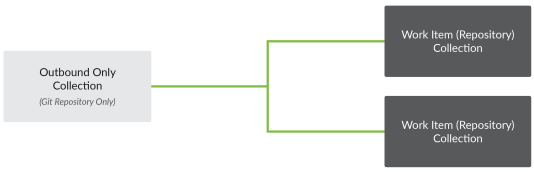
The Code Traceability: Update Existing Work Item template allows developers working in Source Control Management tools, such as Git, to flow data from code commits to an Agile Planning tool, such as Jira, to record information from the commit message directly on the related defect or feature.

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

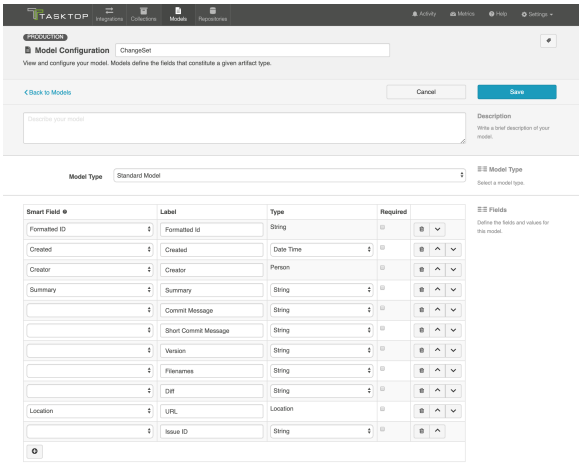
Please review the following pages to learn how to configure your repository:

- 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.

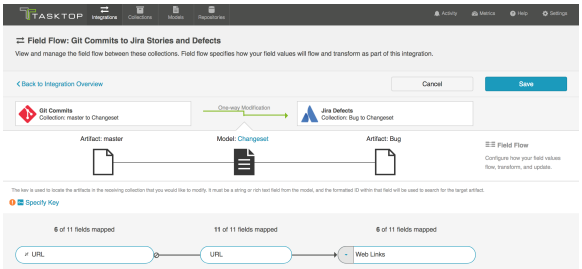
- To configure your source collection (i.e., your Git collection): Outbound Only Collection
- To configure your target collection (i.e., your Jama or Jira collection): Work Item Collection (Repository)
    - Please also review additional notes below

For your **target collection**,

- Ensure you are using the same model as your source collection (i.e., the Changeset model)

Configure the following mappings:

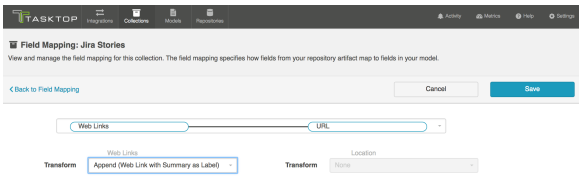
| Source Repository (Git) | Model | Target Repository (i.e. Jama or Jira) |
|---|---|---|
| URL | URL | Web Link |




# 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.



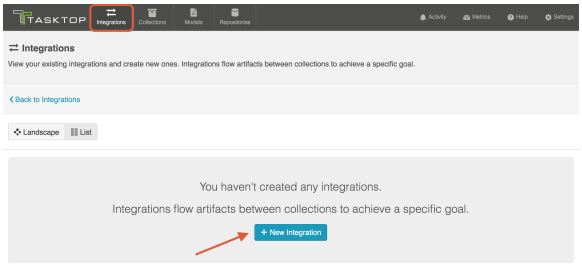

## Important Note about Field Updates

⚠️ 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

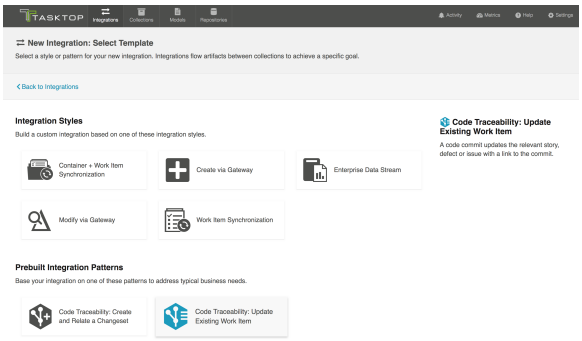
Now that you have all of your base components (i.e. repositories, models, and collections) set up, you can configure an integration to synchronize the artifacts in your collections.

To configure your integration, select **Integrations** at the top of the screen, then click **+ New Integration**.

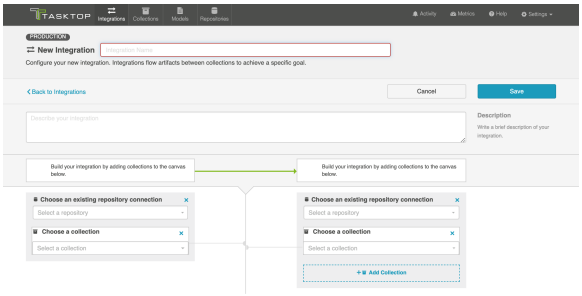



Select the **Code Traceability: Update Existing Work Item** template.

💡 Depending on the edition of Tasktop you are utilizing, you may not have all options available.



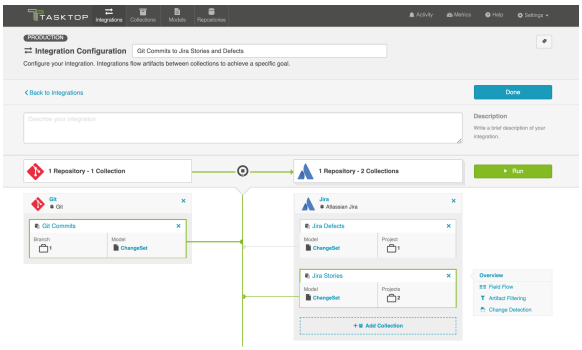

This will bring you to the **New Integration** screen.




Name your integration and select your repositories and collections.

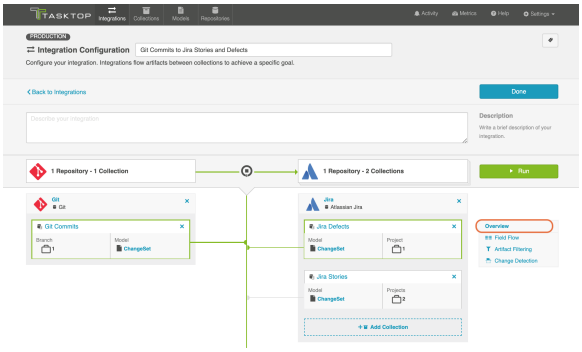
Note that you can add multiple target collections within the same repository if you'd like to flow commit information to multiple artifact types.

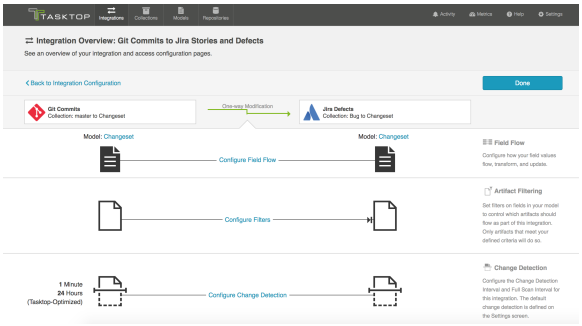
Click **Save**.




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**.

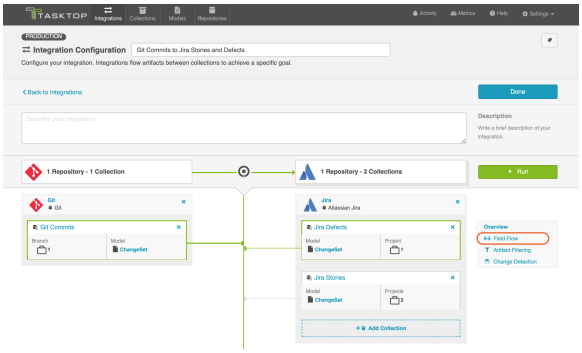




# Field Flow

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.

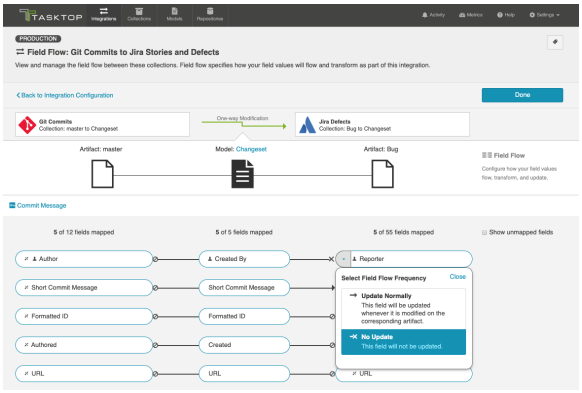


### Important Note about Field Updates

⚠ 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.

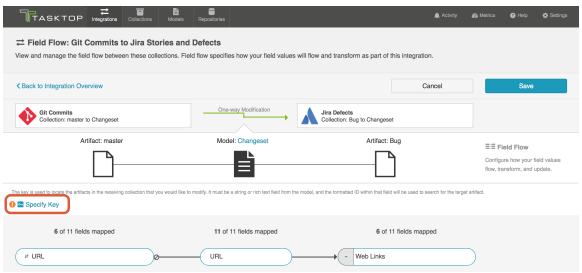


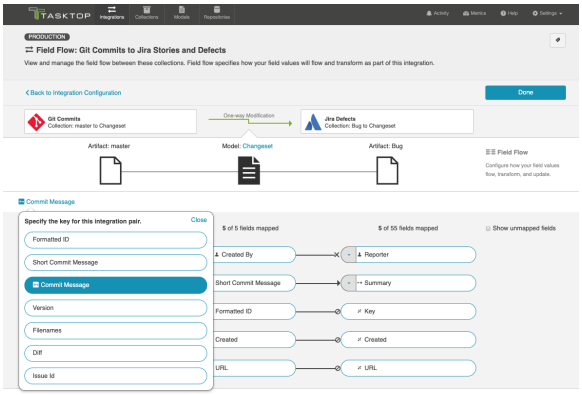

You can see the names of the mapped artifact fields for each collection on the far left and far right, with the model fields displayed in the middle. By default, model fields without mapped repository fields are hidden. You can see all model fields by checking the **Show unmapped fields** checkbox. Constant 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.





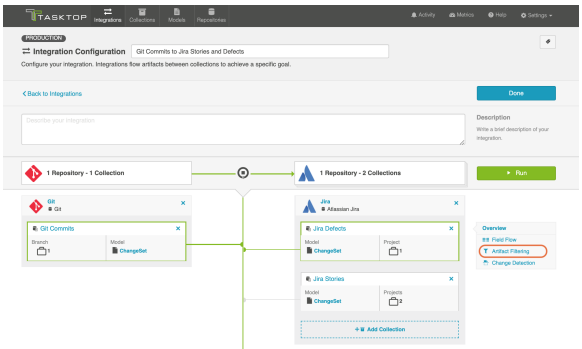

# Artifact Filtering

**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:

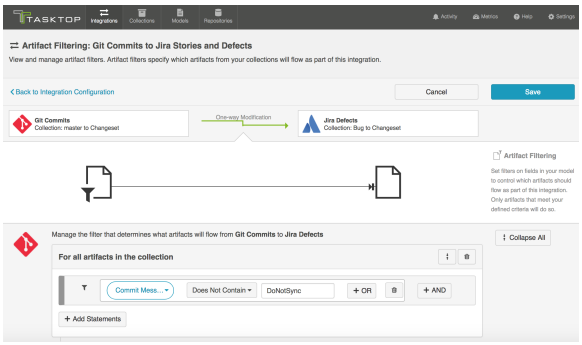
- Be a part of your model, and be mapped to the collection you are filtering from
- Be one of the following field types:
  - Single Select
    - Note that in cases where 'allow unmapped values to flow' is enabled in the model, **only** fields that are already a part of the model will be considered for 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.



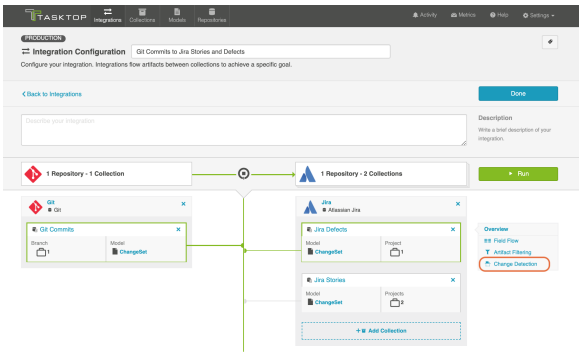

# Change Detection

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.

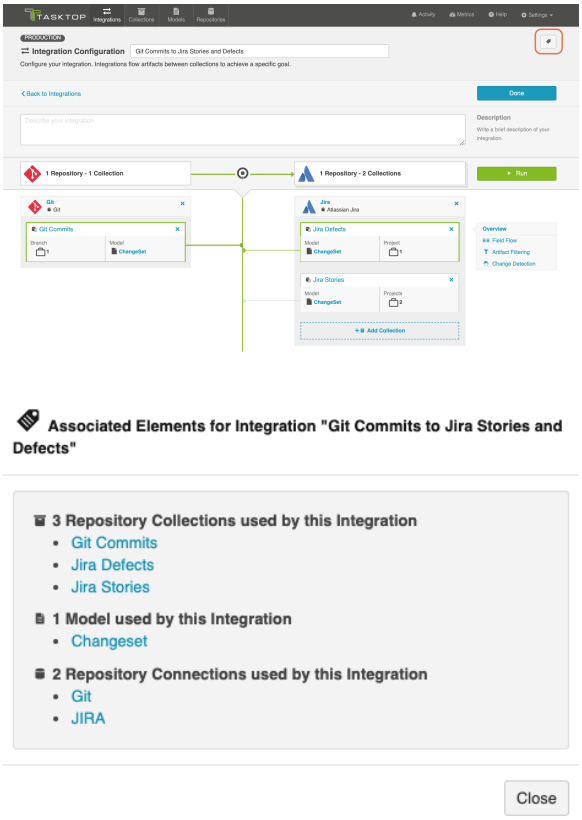
The **Full Scan Interval** is the time between polling requests to detect changed artifacts, in which *all* artifacts of a collection are scanned. Not all changes to an artifact will register as a change. Some repositories do not mark items as changed when (for example) a relationship is added or an 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.




# 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)

# Test Synchronization

*See Tasktop Editions table to determine if your edition contains Test Synchronization functionality.*

| | Supported Repository | Supported Tasktop Version |
|---|---|---|
| | Micro Focus ALM | 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 |

## 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
- ALM Test Steps  Jama Test Steps
- Jama Test Steps  Jama Test Steps

Test Step synchronization is currently supported for the following artifacts:

- Design Steps on ALM Tests
- Run Steps on ALM Test Runs

Use the instructions below to configure the following integrations:

- ALM Test Steps  ALM Test Steps

### Step 1: Connect to your Repository

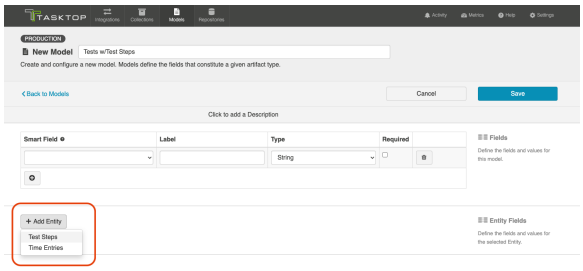
First, connect to your repository by following the instructions here.

You can learn more about ALM-specific configuration in our Connector Docs.

## Step 2: Construct your Model

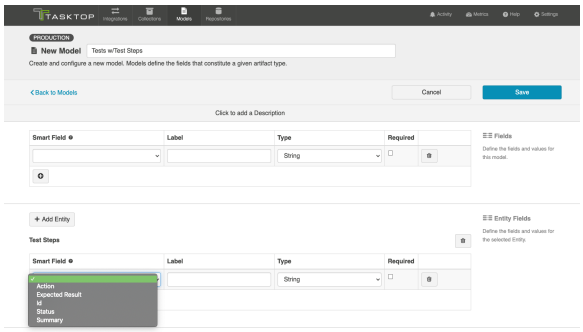
To flow test steps, you will need to add the **Test Steps** entity when creating your Test model.

To do this, click **+ Add Entity** and select the **Test Steps** option.

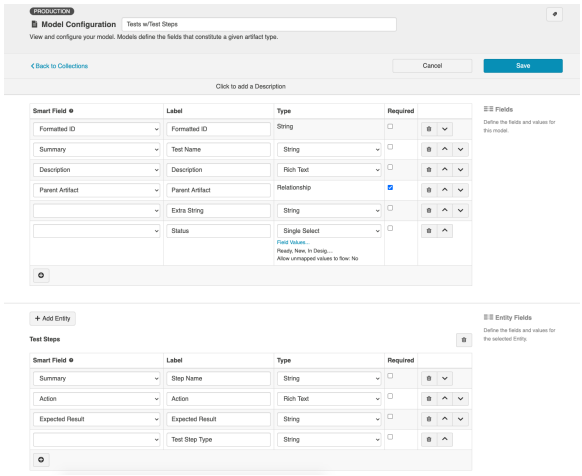



A Test Steps entity will be added. On the Model Configuration screen, you will then see two panels:

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:



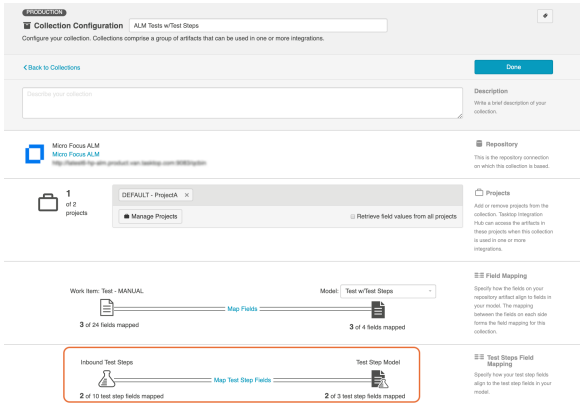

You can find general details on how to create a model here.

# 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.

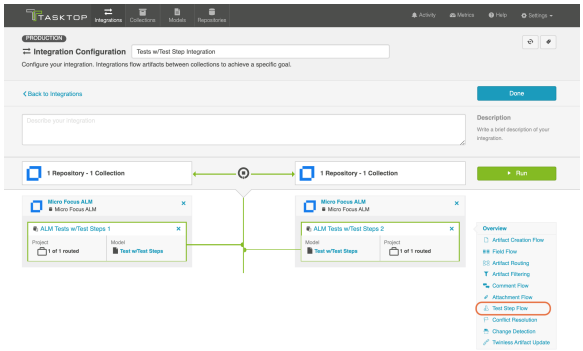
💡 Tests and Test Steps do not require a typical relationship field mapping to link them. We've added behind-the-scenes smarts to couple them for you.

# Step 4: Configure your Integration

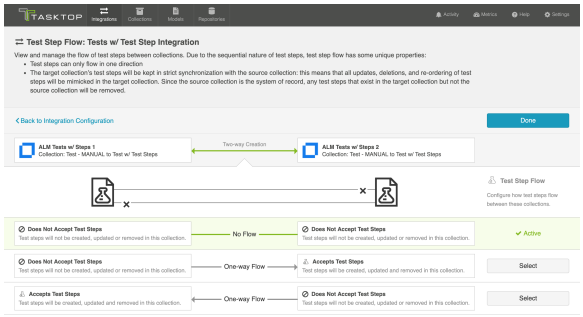
You can find general details on how to configure an integration here.

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:




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.

# 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.

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.

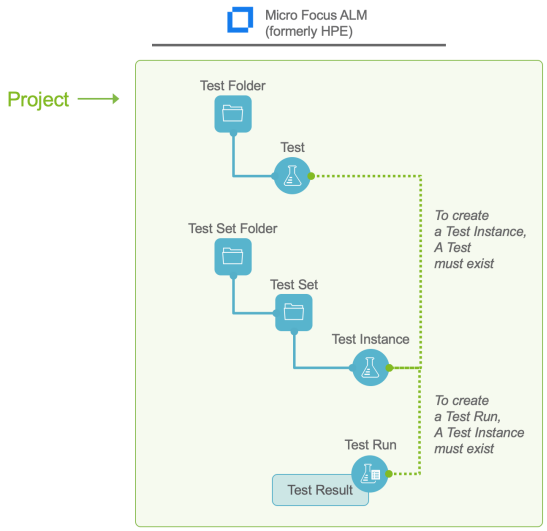
The method outlined below will enable you to flow test results into Micro Focus ALM from Tricentis Tosca or from another ALM instance. Due to the architectural specificity of each external tool, the methods below cannot be used for other endpoints.

You can watch this demo video to learn more:

## Test Architecture

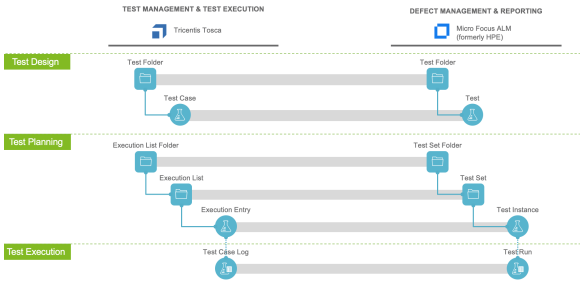
Before you begin configuring your integration, it's important to understand how test artifacts relate to one another.

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!



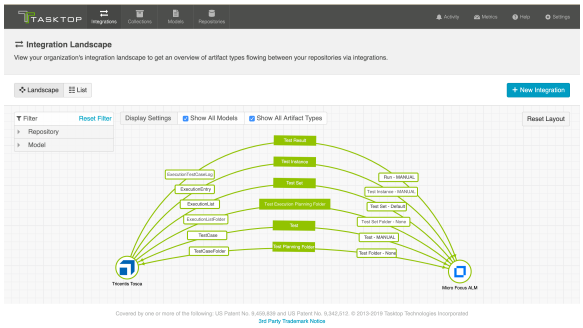

But don't worry — instead of six complex integrations, Tasktop cuts that configuration in half. To set up this integration scenario, you will set up three integrations:

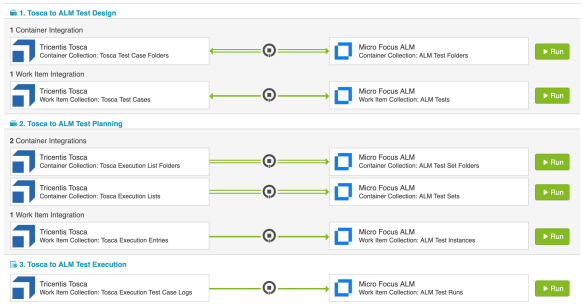
| Integration | Container A | Container B | Work Item |
|---|---|---|---|
| **Test Design** | Test Folder | -- | Test |
| **Test Planning** | Test Set Folder | Test Set | Test Instance |
| **Test Execution** | -- | -- | Test Run |




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:

## Connecting to your External Tools

- Standard Repository Connection

## Creating a Model

- Create or Reuse a Model

## Creating a 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)

## Configuring an 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

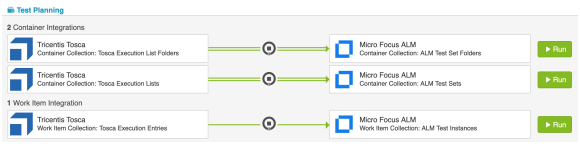
**Supported Artifacts:**

- 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).




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
- Tosca Execution Entries
  - Test Case field must be mapped
  - Execution List field must be mapped

# Step 1: Test Set Folder/Execution List Folders

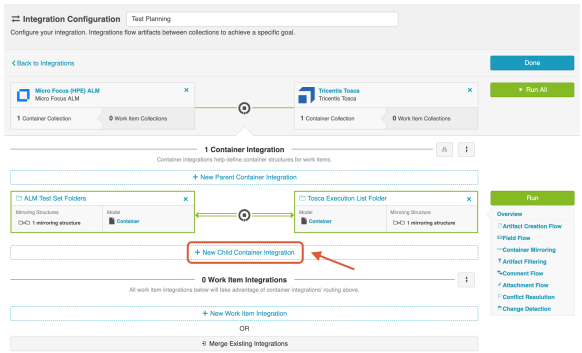
Once your collections have been created and configured, create a Container + Work Item Synchronization. First, configure the top-level container integration:

- 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

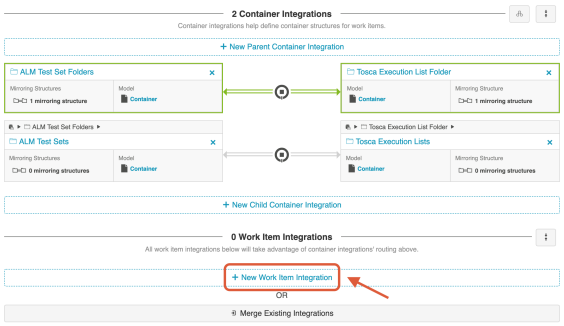
This integration will likely not require container mirroring configuration, as it will inherit that from the parent container integration.

## Step 3: Test Instances/Execution Entry

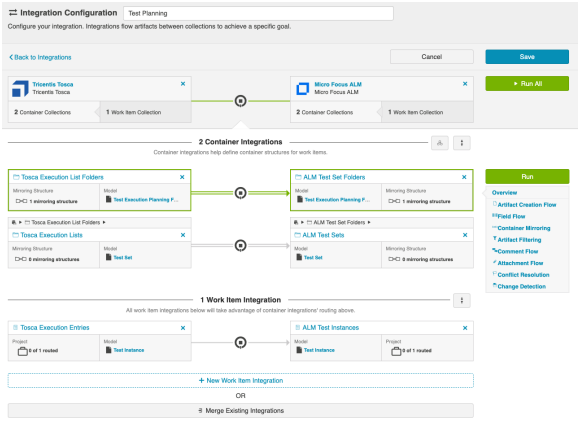
Finally, you will configure your Work Item integration.  It will either be:

- ALM <=> ALM: Test Instance to Test Instance
- Tosca <=> ALM: Execution Entry to Test Instance

Click the 'New Work Item Integration' button to add the integration.



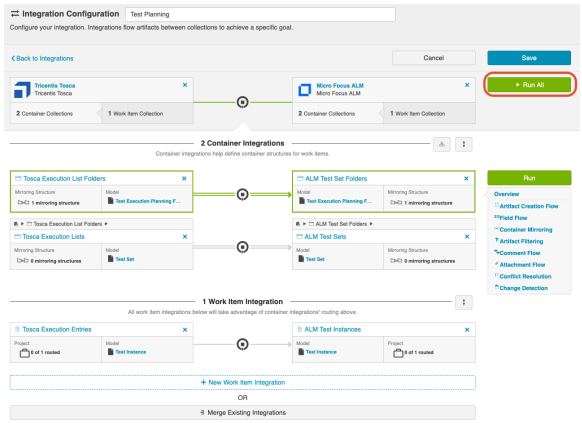

Here is what your fully configured integration will look like:




## 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
- Tosca Execution Test Case Log
  - Execution Entry 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.

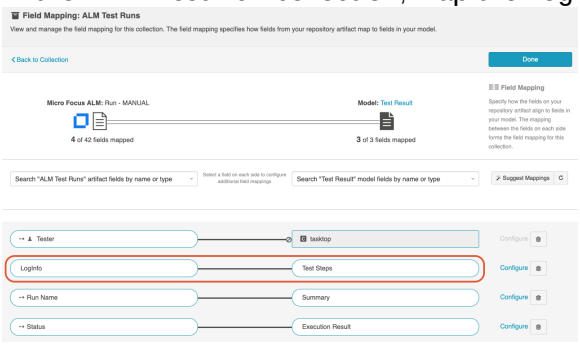
# Flowing Test Step Results in Tricentis Tosca

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 or Tosca  Tosca, it must be configured as part of a Test Execution integration.

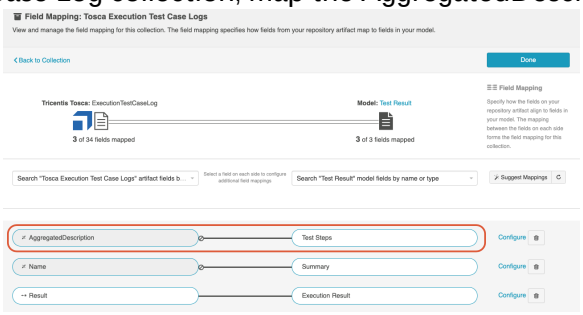
To configure test step flow for Tosca, follow the instructions below:

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.

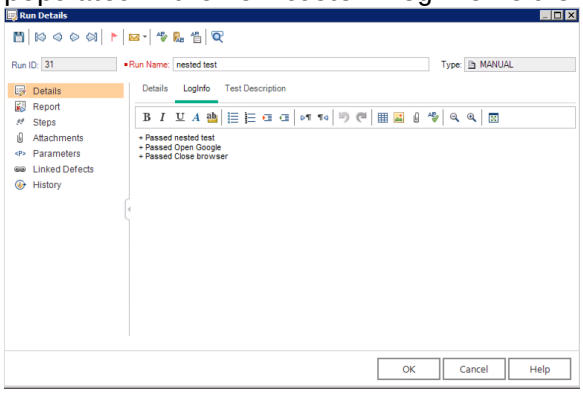
3. In the ALM Test Run collection, map the LogInfo field to the Test Steps field in the model.




4. In the Tosca Execution Test Case Log collection, map the AggregatedDescription field to the Test Steps field in the model.

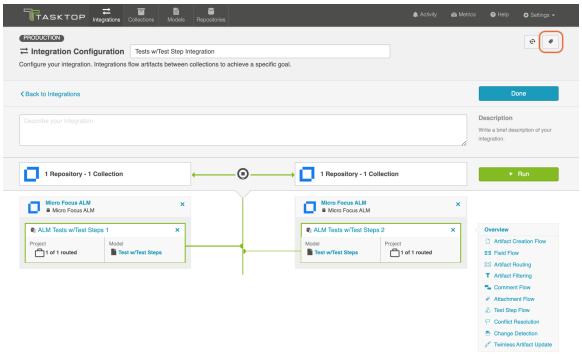



5. Once the integration is run, you'll see that the results of each individual step in Tosca are now populated in the new custom LogInfo field on the ALM Test Run.




# 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

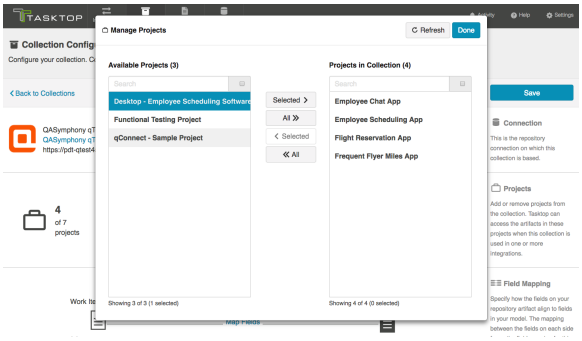
## Expanding the Scale of Your Integration

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

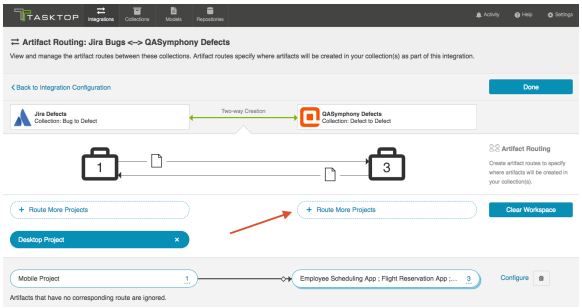
In order to add additional projects from one or more of your repositories to your integration landscape, simply navigate to each collection, and add additional projects as desired.  If you don't see a project you'd like to add, click the 'refresh' button in the upper right corner.



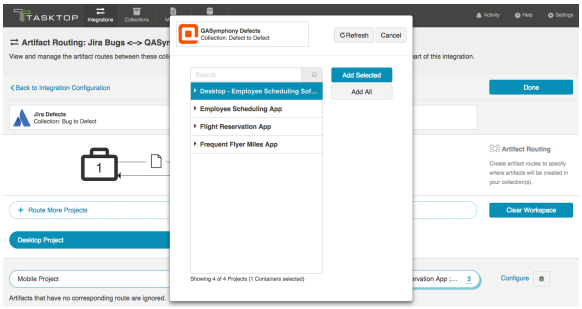

Once your updated projects are saved, navigate to the integration, click on Artifact Routing and route the projects as desired - either creating new routes or adding to existing routes (see instructions below).

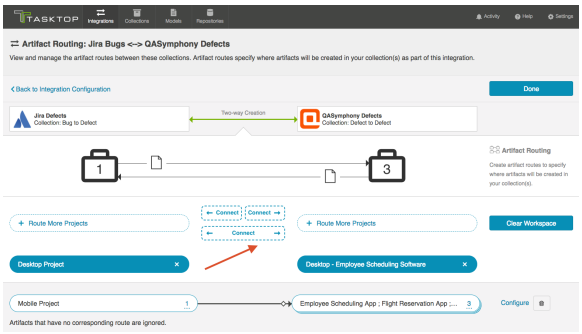
Once the new projects have been added and routed, Tasktop will detect the artifacts contained within the new project(s) at the change detection interval and flow data according to the configuration that you have already set.

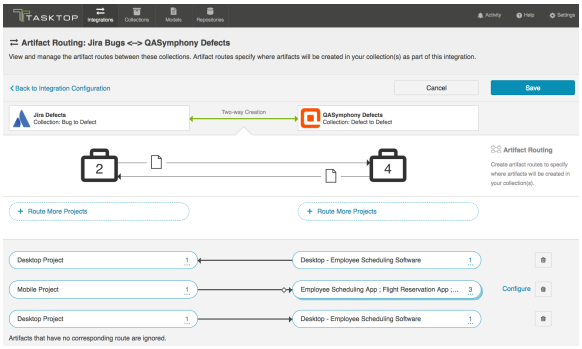
### Add Projects to New Routes:



*Note: If you don't see desired projects, click the 'Refresh' button in the upper right corner.*

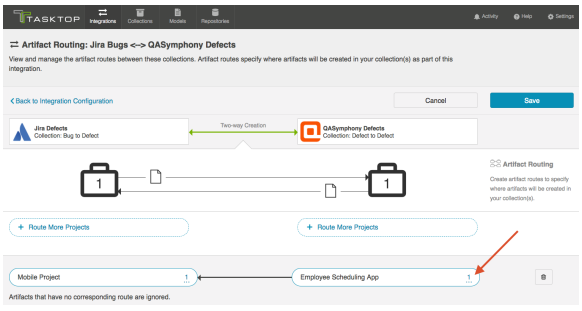


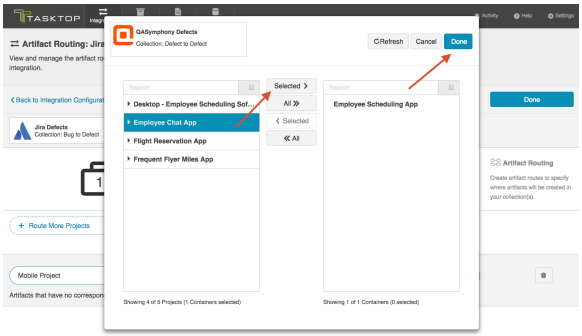

## Add Projects to Existing Routes:

To add additional projects to an existing route, click the numerical link on the right side of the pill.

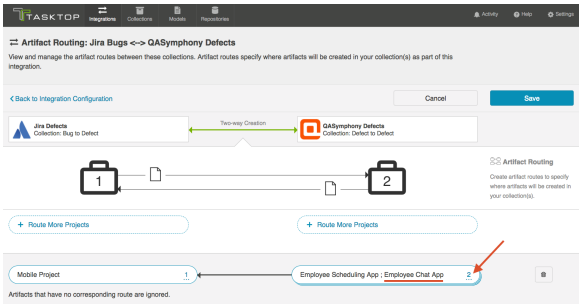



Highlight the project you'd like to add, click 'Selected>' and then 'Done.'

💡 Note: If you don't see the project you'd like to add, click the 'Refresh' button in the upper right corner.



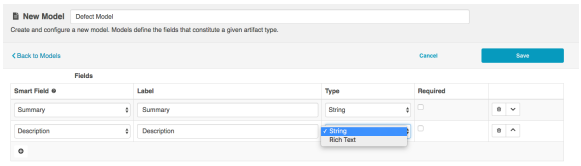

You will now see the updated number of projects, and the additional project's name listed in the pill:



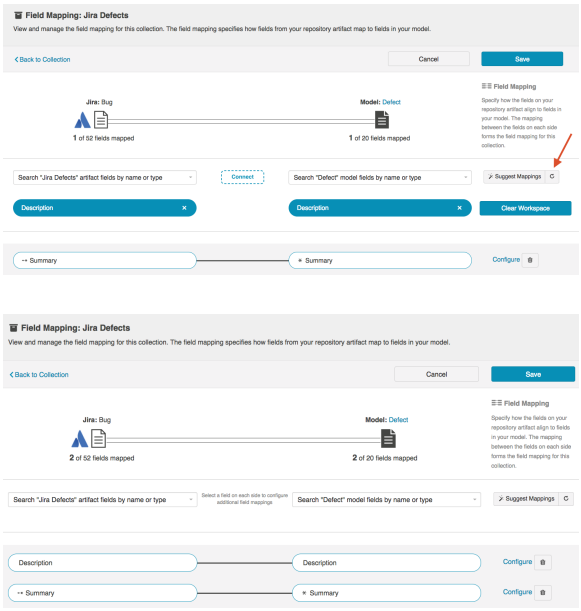

💡 Note: Depending on how you set up your artifact routing, you may need to configure conditional artifact routing.  This will be relevant if you route to more than one target project (as you will need to identify criteria by which the integration can determine which project to flow the artifact to).  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

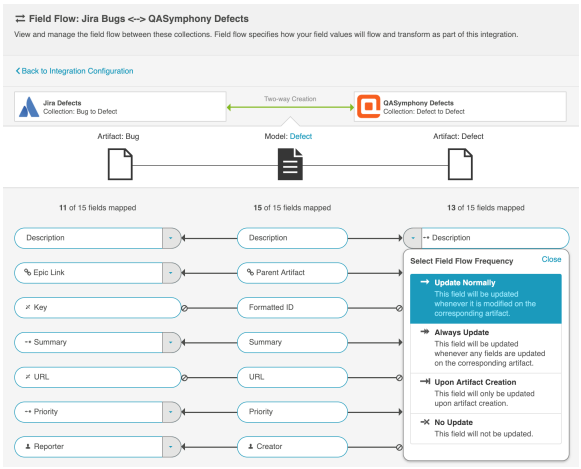
If you'd like to add, remove, or change a field mapping, Tasktop allows you to do so even after the integration has been run.  To add a new field, first make sure it's accounted for in your model.  If needed, you can add a new field on the Model screen:




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

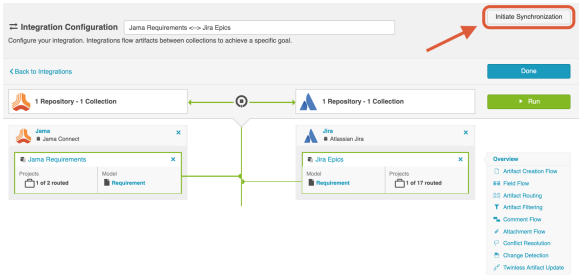
(HPE) ALM, which requires moving projects from an instance running an old version of ALM to a server running a newer instance of ALM. To learn how to move routes from one integration to another, see here.

Since modifications made to existing routes in a running integration can impact internal artifact associations, please contact Tasktop Support before making such modifications.

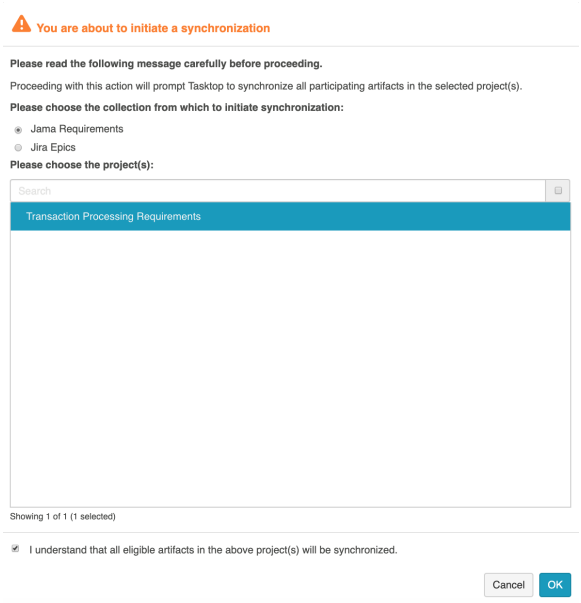
## Expanding Artifact Filters

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.

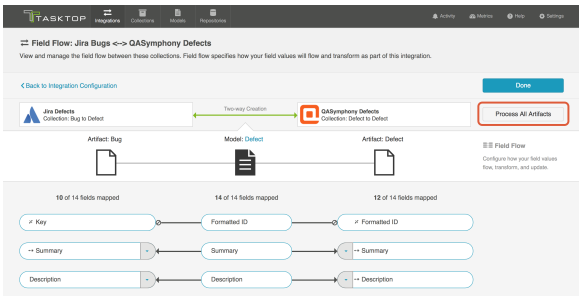
# Processing All Artifacts

⚠ Please contact Tasktop Support before using this feature to ensure you understand its impacts.

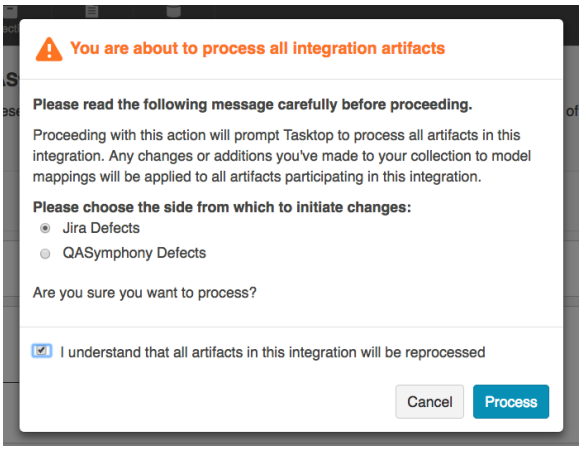
If you'd like force through updates for all artifacts in a collection of your integration, you can click the 'process all artifacts' button on the Field Flow screen.  This can be useful if you add a new field mapping to your configuration or if you change your artifact routing or artifact filtering criteria 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.

# Troubleshooting

## Overview

Tasktop provides several methods for troubleshooting your integration — from our easy-to-use Activity screen which outlines errors, past activity, and more to our Support and Usage Reports which can be used to troubleshoot issues with our support team and to help track Tasktop usage.

Our Configuration History screen contains up to **six months** of changes that have been made on your general settings or configuration elements (e.g., integrations, models, collections, mappings, etc). Please see the Configuration History page for information on migrating these changes from one Tasktop instance to another.

On the Activity Screen page, you can learn about:

- Troubleshooting configuration and licensing issues
- Understanding pending and processing activity
- Reviewing and resolving errors
- Tracking past activity

On the Specific Error Messages page, you can:

- Search for specific errors and review the steps to resolve them
- Learn about in-application error messages

On the Support and Usage Reports page, you can:

- Learn how to download Support and Usage Reports to help troubleshoot issues with Tasktop Support
- Understand the contents of the Support and Usage Reports
- Learn how Tasktop tracks usage information
- Learn how to update your logging settings

Our Error Message Appendix provides a complete list of error messages contained in Tasktop Integration Hub. For information on how to resolve specific errors, please see the Specific Error Messages page, our FAQ, and our Connector Docs (for connector-specific errors).

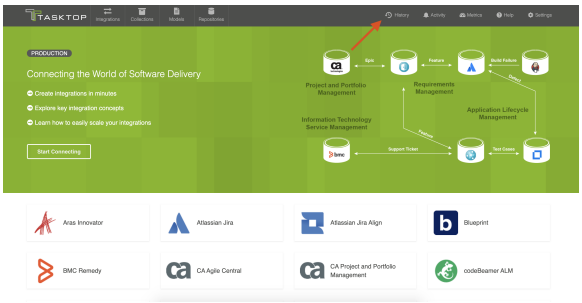
Our Metrics Dashboard provides information on total artifacts created by Tasktop and total artifacts updated by Tasktop, along with a graphical view of the data over time. The dashboard can be used to help troubleshoot Tasktop downtime.

# Configuration History

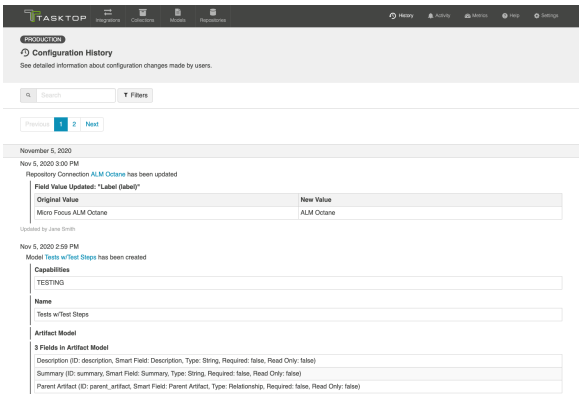
## Introduction

The Configuration History screen shows you up to **six months** of changes that have been made on your general settings or configuration elements (e.g., integrations, models, collections, mappings, etc).

To view your **Configuration History**, click the History icon in the upper right corner.




On this screen, you can see detailed information about the configuration changes and filter these changes by name, date, and the user who initiated the change.




**Tip:** To easily view an updated configuration element, just click on the hyperlink on the element.

## Migrating Configuration Changes

This feature allows you to export configuration changes and migrate them to another Tasktop instance. For example, during testing or major upgrades, changes made in a test environment may need to be replicated in the production environment. Manually replicating these changes is often tedious and time-consuming.

With this feature, you can easily move configuration changes from one instance to another in just a few clicks.

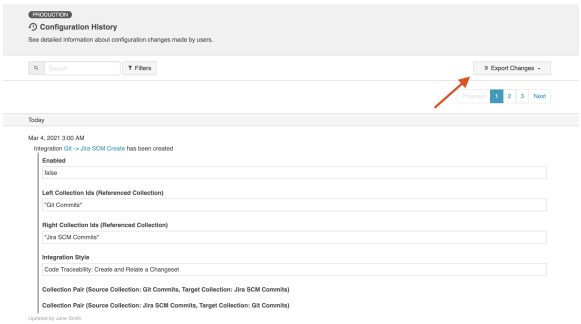
**Before you Begin**

Before using this feature, please review the following requirements:

1. The source environment version should match the target environment version.
2. The source configuration element should mirror the target configuration element as closely as possible.
    a. 💡 **Note**: Changes are applied based on the names/labels of configuration elements in Hub. For example, if you export a change that adds Artifact Filtering to an integration named **Jira to ServiceNow**, upon import Tasktop will search for an integration named **Jira to ServiceNow** and apply the change.

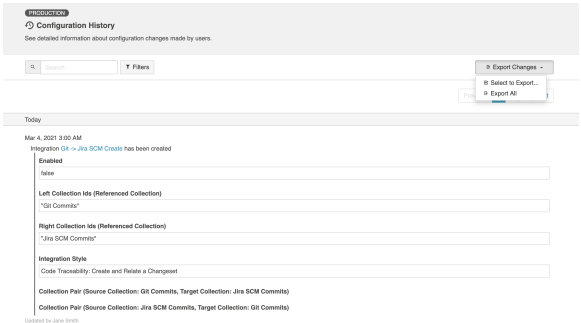
# Exporting Changes

To begin exporting configuration changes, click **Export Changes.**

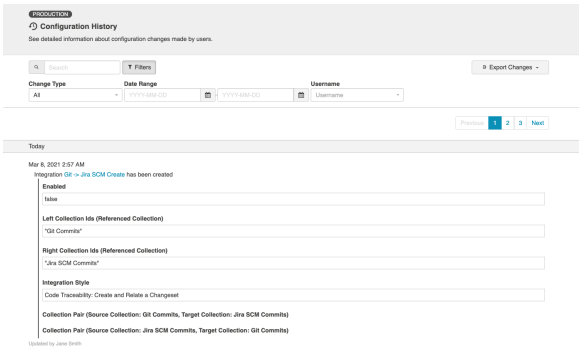



Click **Select to Export** to export selected changes or **Export All** to export all configuration changes.

💡 **Note**: Any changes made on the **Settings** screen cannot be exported (e.g., adding a license or updating the change detection interval). Additionally, any credentials will not be exported and will need to be re-entered upon import.

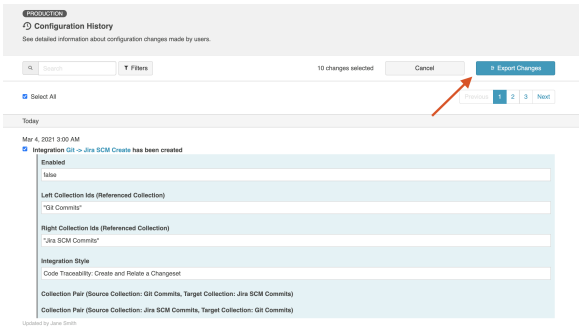



If you'd like to narrow down changes, you can filter by type, date range, and user.



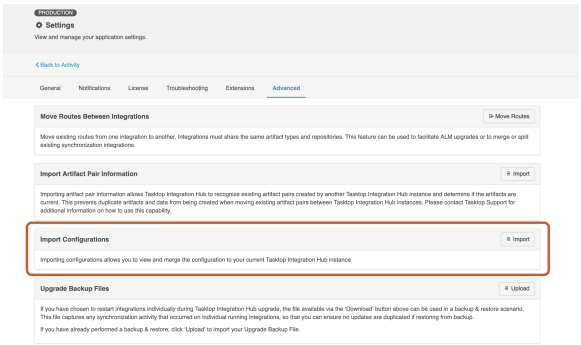

After you've selected the desired changes, click **Export Changes** and a .**zip** file will be generated with the changes.

💡 **Note**: If you attempt to modify the contents of the **.zip** file, you may encounter issues upon import.

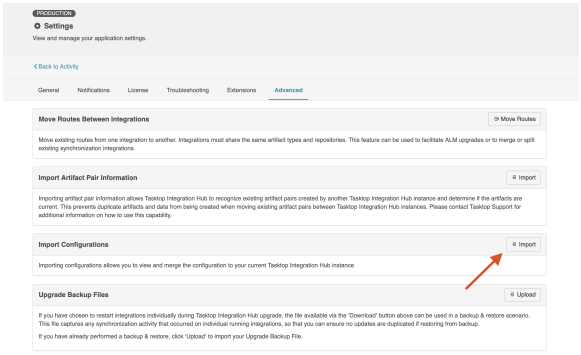



# Importing Changes

You can import the exported changes in the **Advanced** tab on the **Settings** screen.

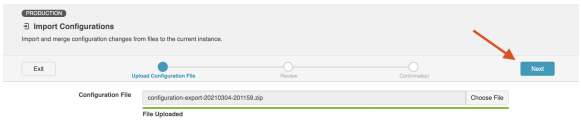



Click **Import** in the **Import Configurations** section.

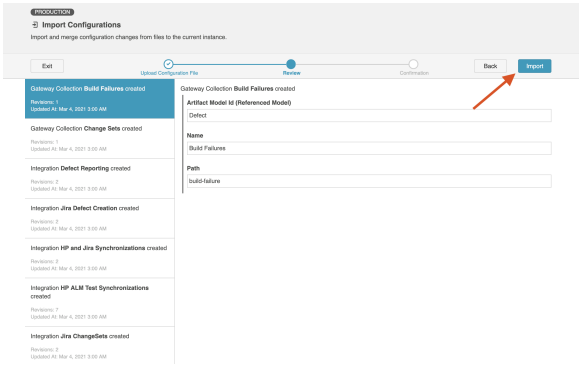



On the **Import Configurations** screen, select the **.zip** file with the changes you'd like to import.


Click **Next** to review the changes before importing.



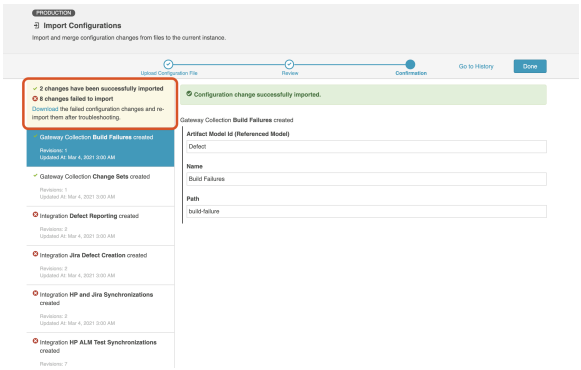

After reviewing, click **Import** to apply the changes.



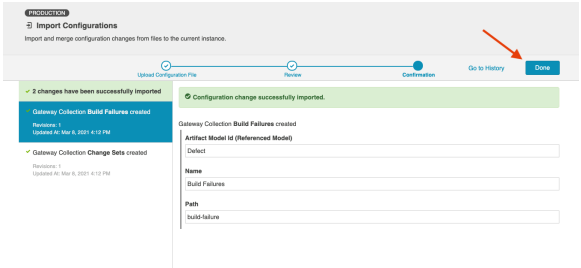

If a change fails to import, an error message will appear to explain why the change failed. Once a change fails to import, Hub will not attempt to import the subsequent changes and will provide a new **.zip** file containing the remaining changes to import after troubleshooting the failed change.

**Possible causes of import failure:**

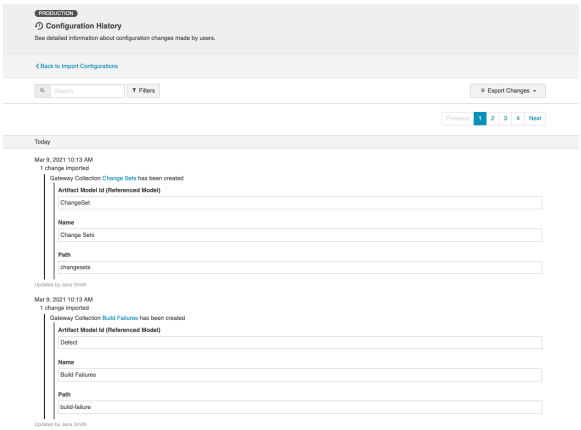
- **Missing element in the target Hub instance**. This error can be resolved by manually creating the missing required element in the target system or importing the changes that created that element from the source Hub instance to the target Hub instance.
- **Element already exists in the target Hub instance**. This error can be resolved by deleting the element, or not exporting the changes that try to create the duplicate element.




Once you're finished importing configuration changes, click **Done**.



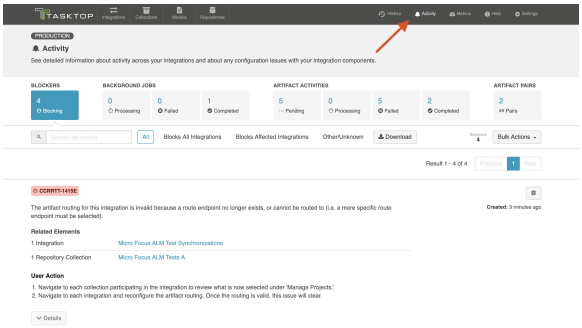

The imported changes will be visible on the **Configuration History** screen.



# Activity Screen

## Introduction

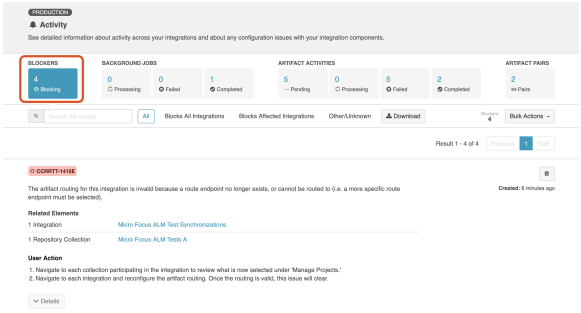
Most problems can be solved by navigating to the **Activity** screen and following the steps described on the listed errors. The Activity screen can be accessed by clicking **Activity** in the top right corner of the screen.




## Blockers

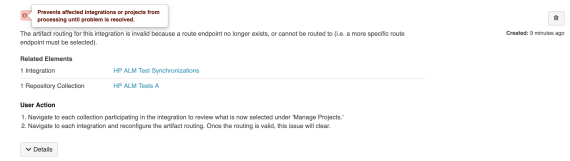
The **Blockers** tab shows issues that arise from **invalid Tasktop configuration** or from more **global issues,** such as having an **invalid or expired license**. These are issues that can generally be resolved within the Tasktop application itself.

💡 **Tip**: Blockers can block integrations from running, so it is recommended that you monitor the Blockers tab regularly.

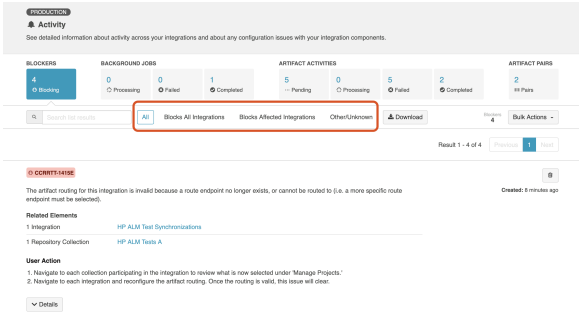



An additional warning icon appears when blockers are so fundamental that they will prevent integrations from running.

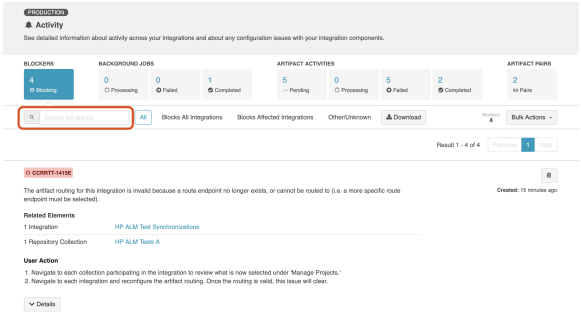
The hover message will indicate whether the blocker will prevent **all** integrations from running (e.g., licensing errors), or just **affected** integrations from running (e.g., a configuration error that impacts just one integration).



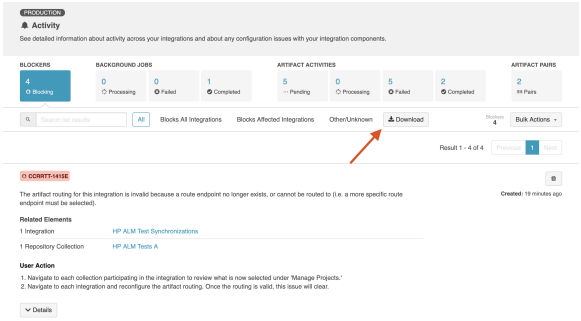
You can filter based on blocker impact using the filters at the top of the screen.



Or, you can use the search box to refine your results.



Click **Download** to export a .**csv** file containing your Blockers.



You can take the following actions on the Blockers tab:

- **Retry**: Retries a blocker. This action is only available for configuration migration blockers.
- **Resolve:** Resolves a blocker. This action is only available for certain blocker types, and can be taken to acknowledge that the user has reviewed the blocker and taken any required user actions.

- **Remove**: Removes a blocker. If the blocker is blocking an integration, the integration will become unblocked. However, if the cause of the blocker has not been resolved, the blocker will return to the Blockers tab the next time configuration validation occurs (once an hour).

You can also take the following Bulk Actions:

- **Refresh**: Refreshes the blockers tab.
- **Remove All**: Removes all blockers. If the blockers were blocking an integration, the integration will become unblocked. However, if the cause of the blocker has not been resolved, the blocker will return to the Blockers tab the next time configuration validation occurs (once an hour).

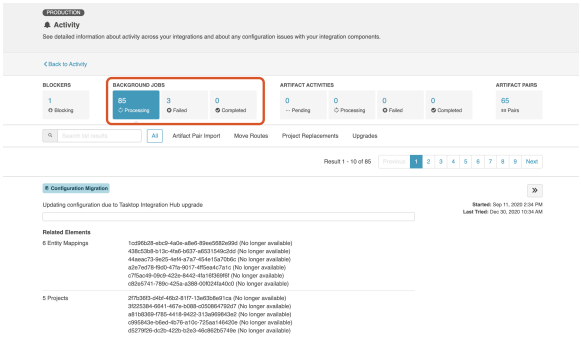
# Background Jobs

The **Background Jobs** section shows progress on background Tasktop processes such as: Upgrades, Red eployments from Sync, Project Replacements for invalid projects in Tasktop collections, and Moving Routes between Integrations.

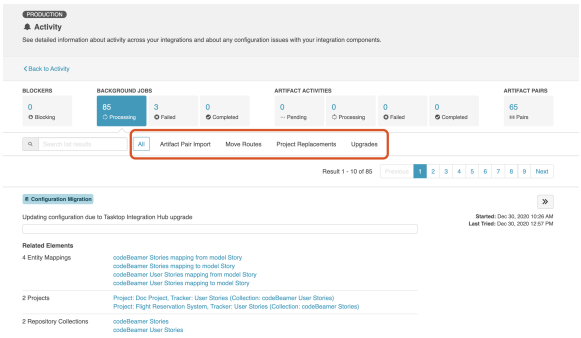
Background Jobs consists of three subcategories:

- **Processing**: Background Jobs that are currently processing
- **Failed**: Background Jobs that Tasktop attempted to process, but were not successful
- **Completed**: Background Jobs that have successfully completed

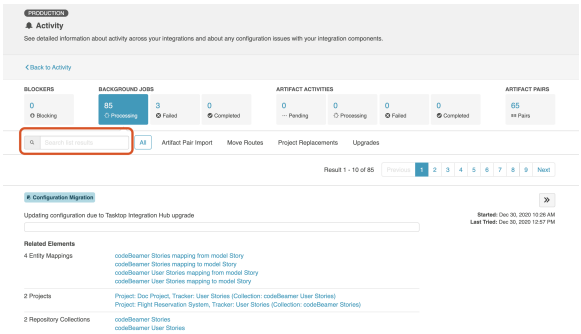
You can take different actions on the background jobs in these subcategories, which are outlined in the sections below.




You can filter based on job type for each category.




Or, you can use the search box to refine your results.



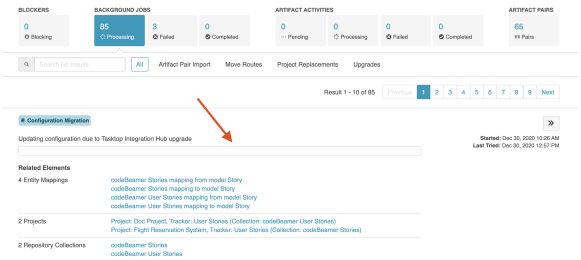

# Processing

In the **Processing** tab, you can take the following actions:

- » **Prioritize**: Prioritizes the processing background job in the queue

While background jobs are processing, you will see a progress bar to track progress.

💡 **Note**: Jobs that are in progress cannot be canceled.




# Failed

In the **Failed** tab, you can take the following actions:

- » **Prioritize**: Prioritizes the retry of the background job in the queue. This option is especially useful if you have made changes in your repository or in Tasktop that will likely clear up the failed job.
    - You will see this action if the event is already set to be retried, and is hence both in **failed** and **processing** states simultaneously.

If a background job fails, it will appear color coded in red. If there is an associated issue, a link will be shown to navigate to that issue. These jobs will be retried automatically until they complete, and can be prioritized using the **prioritize** button.


# Completed

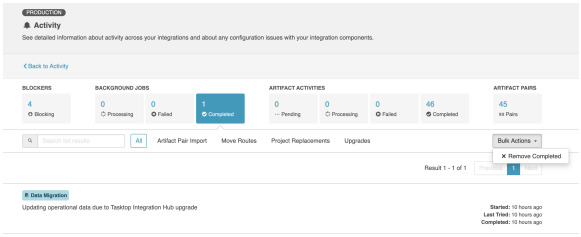
In the **Completed** tab, you can take the following actions:

- ✖ **Remove Completed**: Removes all completed background jobs.

Once jobs complete, you will see them in the **Completed** tab of the Background Jobs section color coded in green. For Project Replacement jobs, you can expand the **Projects Updated** section to see additional details:


You can remove all completed background jobs using the Bulk Actions dropdown.

💡 **Tip**: Activity listed on the Background Jobs tab will be cleared after each Tasktop upgrade.



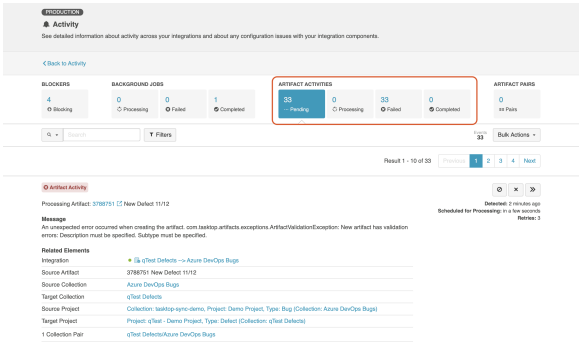

# Artifact Activities

The **Artifact Activities** section shows **activities that are active in an integration.**

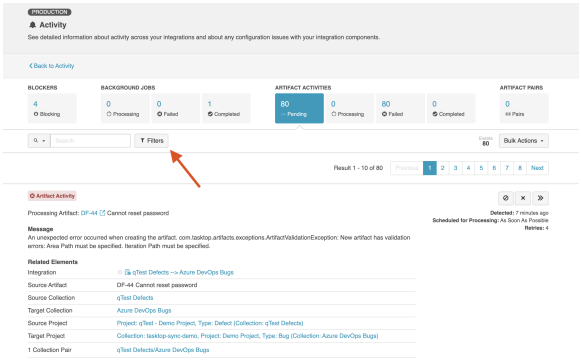
Artifact Activities consist of four subcategories:

- **Pending**: Activities that are queued up to be processed.
- **Processing**: Activities that are currently processing.
- **Failed**: Activities that Tasktop tried to process, but was not successful.
- **Completed**: Activities that have completed processing.

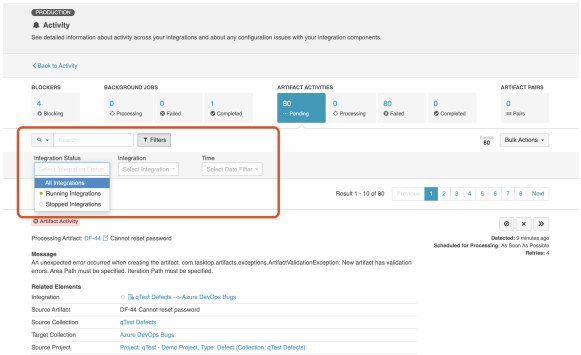
You can take different actions on the events in these subcategories, which are outlined in the sections below.



Each category allows you to expand your filter options.

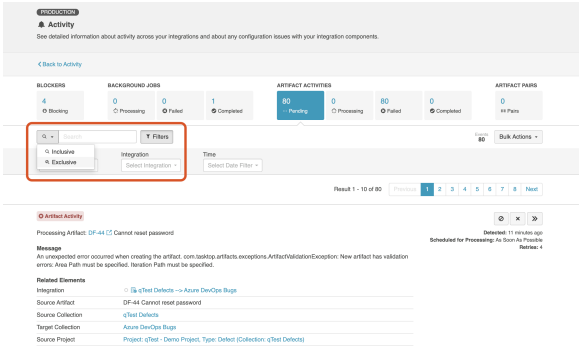


You can filter by search, integration status (e.g., running or stopped), integration name, or created date.

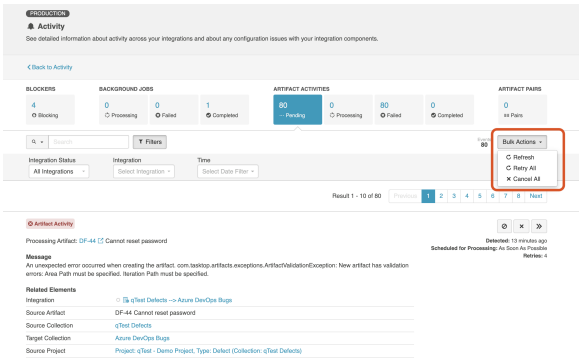


You can also filter to exclude specific text.

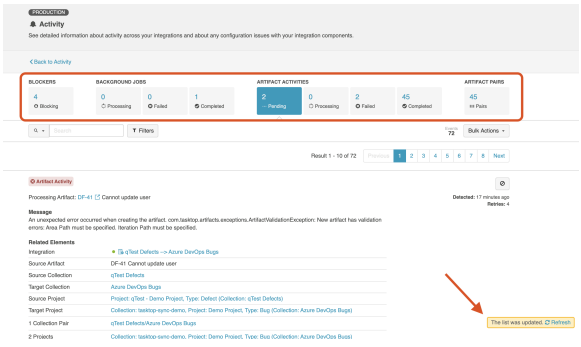
To do this, click the search icon and choose **Exclusive** in the dropdown menu. Specify the text you would like to exclude, and only artifact activities without this text will be displayed.



Each category also allows you to take bulk actions.



The number of events in the summary banner will update regularly, but the list of events themselves will need to be refreshed to show new activity. This is to avoid items unexpectedly appearing and disappearing when you might be examining them.



# Pending

On **Pending** Activity, you can take the following actions:

- **Prioritize**: Prioritize this pending event in the queue.
- **Cancel**: Remove this event from the pending queue. It will not be processed, though subsequent changes to artifacts will trigger another event.
- **Ignore**: If an error is pending, you have the option of moving it to the Ignored Errors tab.  See Errors section for details.

# Processing

The **Processing** tab shows activity that is currently processing. There are no actions that can be taken here.

# Failed

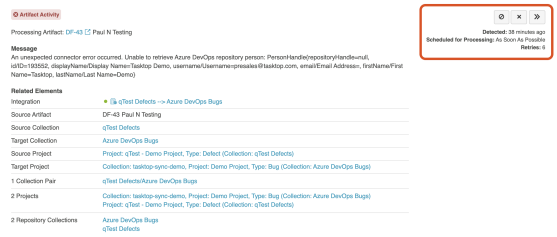
The **Failed** tab shows any failed activity related to specific activities that have occurred. In contrast to the Blockers tab, failed activity here typically blocks **individual artifacts** rather than **entire integrations**, and therefore are less severe.

You can take the following actions:

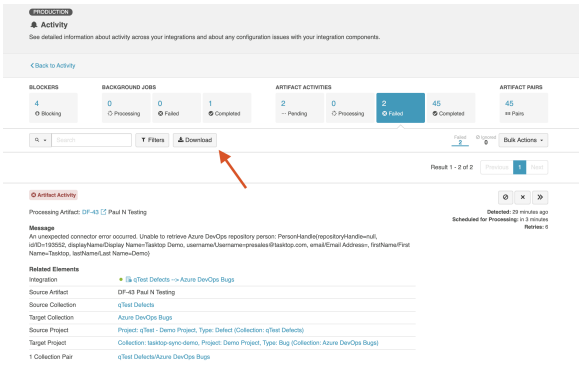
- ⊘ **Ignore**: Moves the failed activity to the **Ignored** list. Once ignored, it will no longer show up in the Failed list (or in Pending), and it will not be counted in the Failed summary counts at the top of the screen.
- ✖ **Cancel**: Removes the failed activity from the list. It will not be retried, though subsequent changes to artifacts will trigger another event.
- » **Prioritize**: Prioritizes the retry of this failed activity in the queue. This option is especially useful if you have made changes in your repository or in Tasktop that will likely clear up the error.
    - You will see this action if the event is already set to be retried, and is hence both in **failed** and **pending** states simultaneously.
- ✚ **Recreate**: If a previously synchronized artifact has been deleted in one of your repositories, you have the option of recreating it from the Activity screen. This will keep the newly recreated artifact in sync with the source artifact.
- ↻ **Retry**: Retries the failed activity.
    - You will see this action if the event is not already set to be retried.


💡 **Note**: Most failed activity will automatically be retried on a gradually decreasing interval (granted that Tasktop can locate the artifact that is to be changed). Retry-able failed activity will be retried approximately 30 seconds after they are first encountered, and then on a gradually decreasing interval over time.

You can see information about retries on the failed activity itself. In the example below, you can see that the failed activity has been retried 6 times, and that it has been scheduled for processing as soon as possible. If a failed activity will not be retried, this information will not be be relevant and hence will not be displayed.




Click **Download** to download a .**csv** file containing your failed artifact activities.



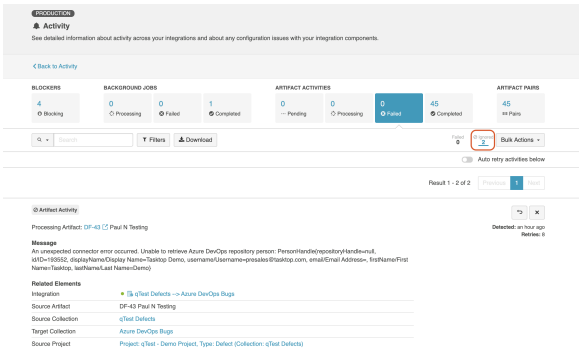
A complete list of failed artifact activities (i.e., errors) is available in the appendix.

You can find additional information on select errors in our FAQ.

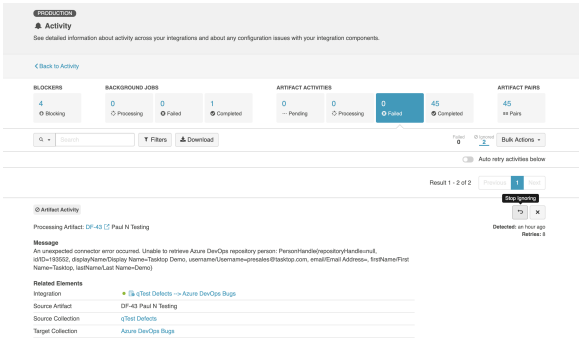
## Ignored

If you ignore a failed activity, it will be moved to the **Ignored** list, and no longer be counted in the **Failed** total at the top of the screen.

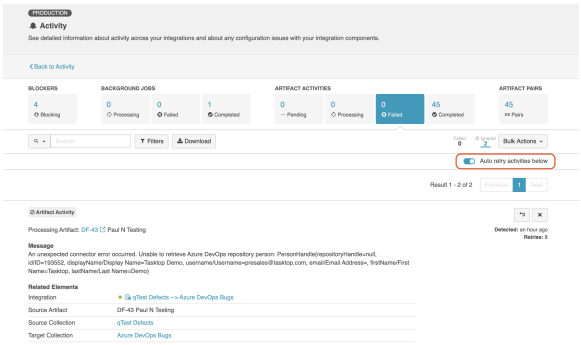
💡 **Note**: Ignored artifact activities must be manually retried to be resolved.



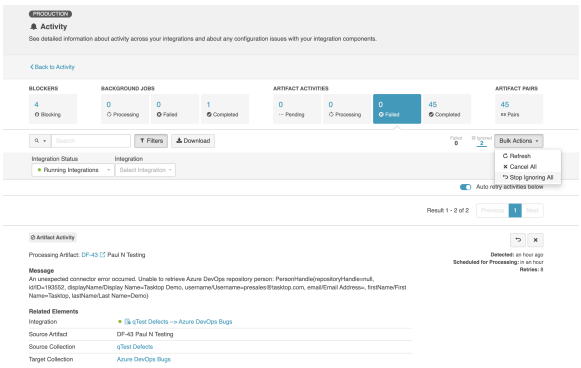
You can move a failed activity back to the Failed list by clicking **Stop Ignoring**.



If you enable **Auto retry activities below**, all ignored artifact activities will be retried automatically.



If you'd like to use the bulk action, **Stop Ignoring All**, you must first apply a filter to the Ignored list. This will move all failed activities that meet your search filters back to the Failed list.

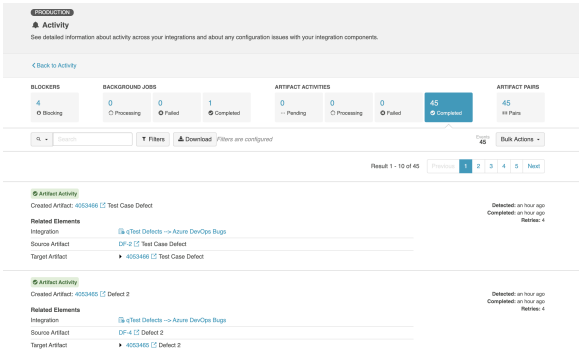


# Completed

The **Completed** tab allows you to view all past integration activity, so that you can understand **what has successfully completed**.

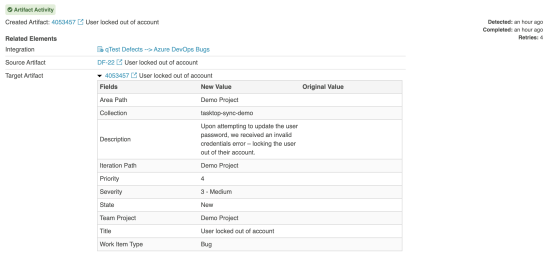
There are three types of Completed activities:

- **Created Artifact**: When a new target artifact is created in a repository
- **Updated Artifact:** When an existing artifact is updated in a repository
- **Associated Artifacts**: When existing artifacts are auto-matched, and therefore associated with one another. Currently this is only supported for containers, when utilizing Container Matching fo r a Work Item + Container Mirroring synchronization integration.



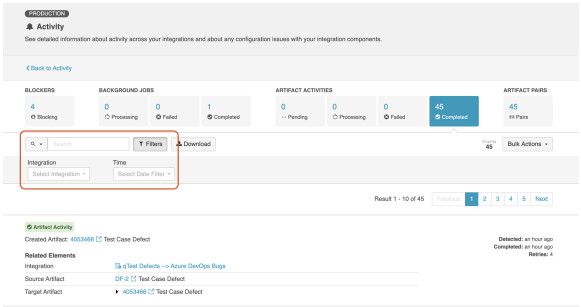
You can click the drop down arrow on each activity to see more details on the activity that has occurred.

If past activity is indicating that a new artifact was created, you'll see that the Original Values listed are blank, and that the Activity type is **Created Artifact** as opposed to **Updated Artifact**.

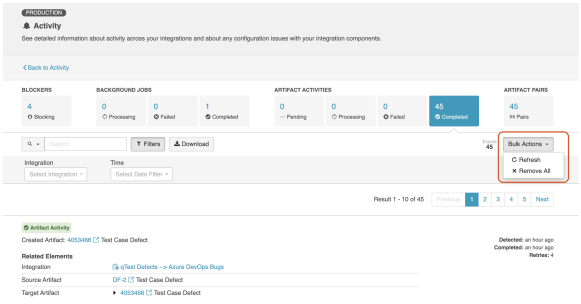


If you'd like to filter your results, you can use the search box.

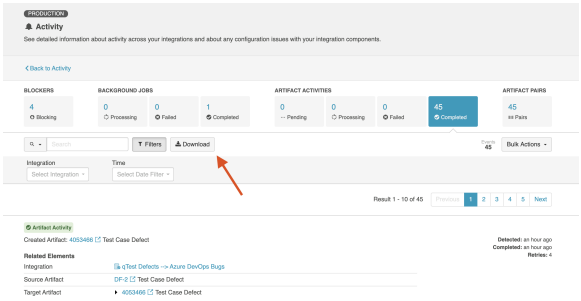
Additionally, you can click **Filters** to expand filtering options. You can use the integration filter to filter by integration, or the date filter to filter by a fixed date range or by a set number of days in the past (which will dynamically update your results as days pass).



You can use **Bulk Actions** to refresh, or remove all artifact activities that meet your filters. If you have not configured any filters, all completed artifact activities will be refreshed or removed.



Click **Download** to download a .**csv** file containing your completed artifact activities.



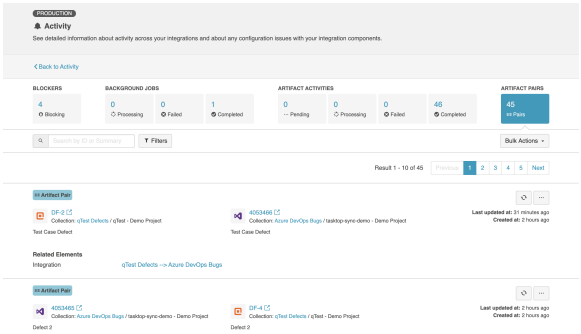
⚠️ **Note**: Tasktop will store up to 100,000 entries on the Completed screen. Once 100,000 entries are met, older entries will be deleted as new entries come in. You can also opt to clear your entries when approaching 100,000 to have better visibility into more recent completed artifact activities.

# Artifact Pairs

> ⚠️ If using a Keycloak installation, please follow the instructions here to enable full functionality of this feature.

The **Artifact Pairs** tab allows you to view and manage artifact associations so you can promptly address problems related to specific artifact pairs.

💡 **Note**: For each artifact that is displayed, there is no directional information involved with the pair (i.e., Tasktop does not display which artifact is source or target) — this tab only shows the artifact association.

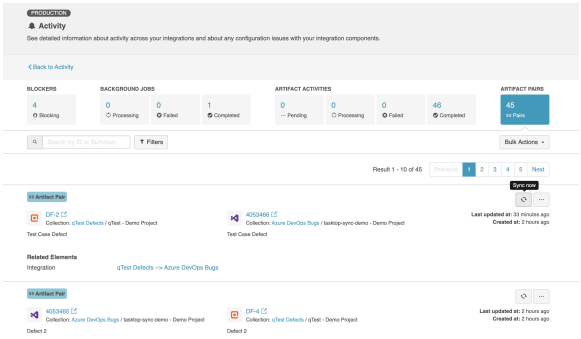


## Synchronizing Artifact Pairs

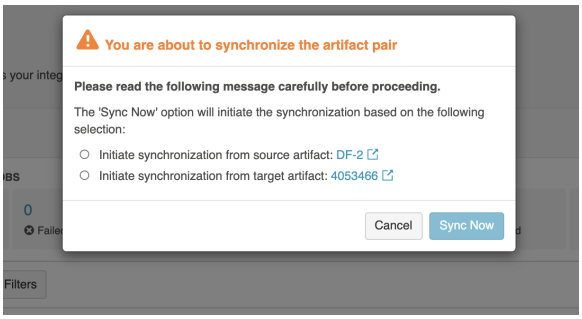
Rather than synchronizing the full collection of artifacts in an integration, Tasktop allows you to synchronize a **single artifact pair**.

Click **Sync now** to synchronize an artifact pair.

💡 **Tip**: If you'd like to synchronize **multiple** artifact pairs, refer to the section below.



A pop-up will appear prompting you to select the artifact from which you'd like to initiate the synchronization.



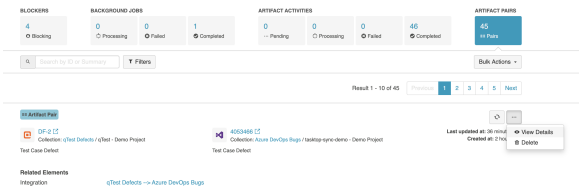

## Viewing Artifact Pair Details

> ⚠ If using a Keycloak installation, you **must** follow the instructions here before you can proceed.

If you'd like to view the details of an artifact pair, click the ellipses and select **View Detail**s.

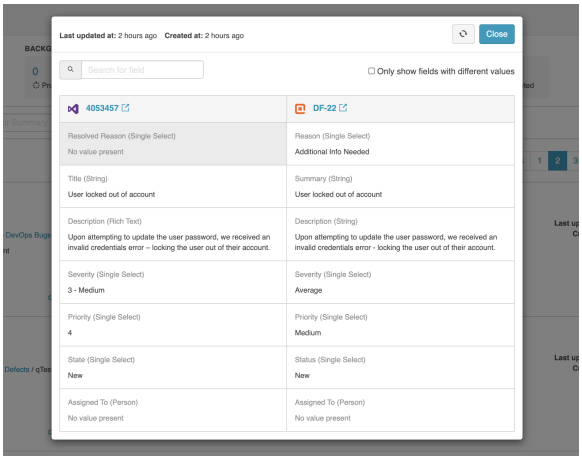
💡 **Tip**: The View Details option may not appear if there is missing information on either side of the artifact pair (e.g., if a collection or repository is deleted on either side of the pair).

⚠ **Note**: The artifact summary may not appear if using a custom string field as the summary field.

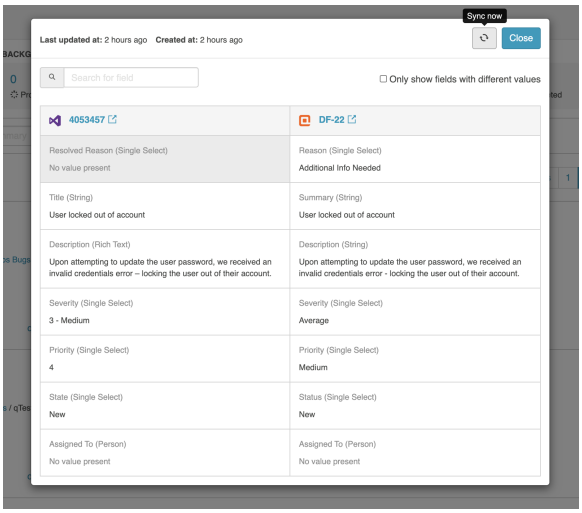


A pop-up will appear with a table of field values belonging to the artifact pair. Any greyed out area represents the source field in a unidirectional mapping.

💡 **Note**: Each row in the table corresponds to a field mapping. For more complex mappings such as one-to-many, each field mapping will remain in a single row.

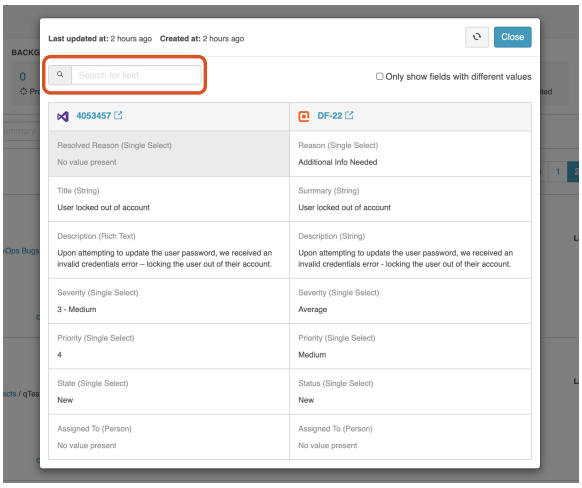



Within the pop-up, you can initiate synchronization of the pair by clicking **Sync Now**.




If you'd like to find a field and many details exist, you can use the search box in the upper left corner to find a specific field.

💡 **Note**: The search option within the pop-up will only search for fields and **not** values.

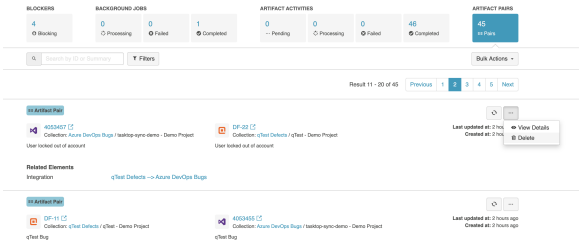


# Deleting Artifact Pairs

> ⚠️ If using a Keycloak installation you **must** follow the instructions here before you can proceed.

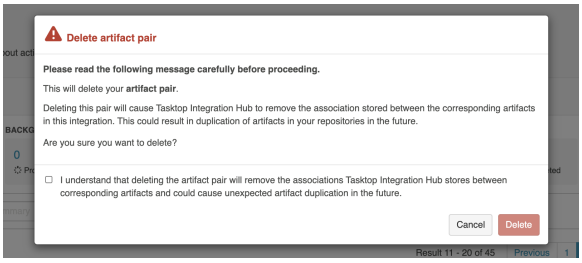
If you'd like to delete an artifact pair, click the ellipses located in the right corner of the artifact pair and select **Delete**.

💡 **Tip**: To delete multiple artifact pairs, refer to the section below.



A pop-up will appear confirming you'd like to proceed.

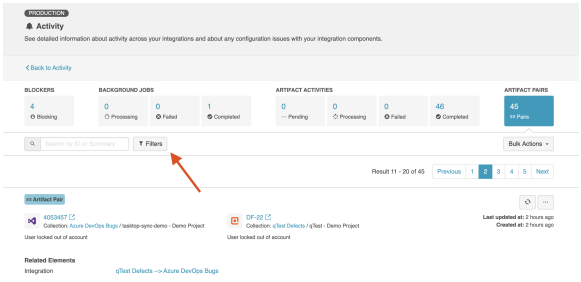
⚠️ **Note**: Deleting an artifact pair will remove all associations from the database, which may result in duplicate artifacts if change detection picks up the deleted artifact again.



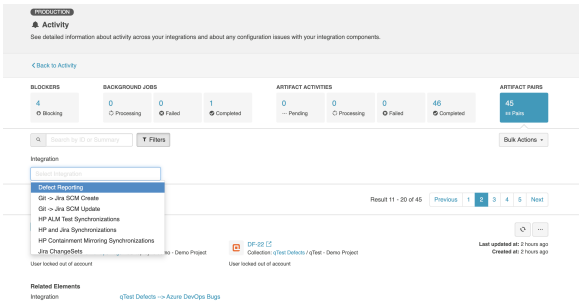

💡 **Note**: Deleted pairs are logged under a file named `deleted-pairs.log` in the Support and Usage Reports. If you're unsure if an artifact pair has been deleted, you can revisit the log file to confirm.

# Filtering Artifact Pairs

If you'd like to filter your artifact pairs, you can use the search box to refine your results by ID or summary. Additionally, you can expand your filter options by clicking **Filters**.



Using expanded filters, you can filter by integration.

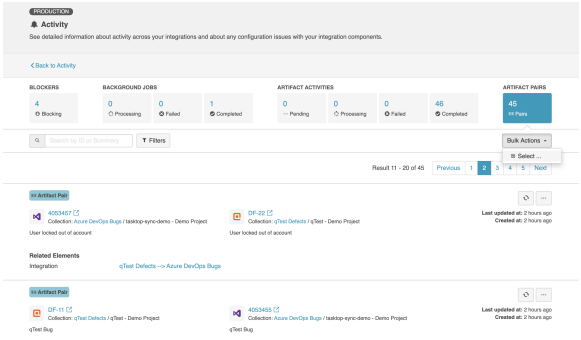


## Bulk Actions

You can also take bulk actions like synchronizing or deleting multiple artifact pairs.

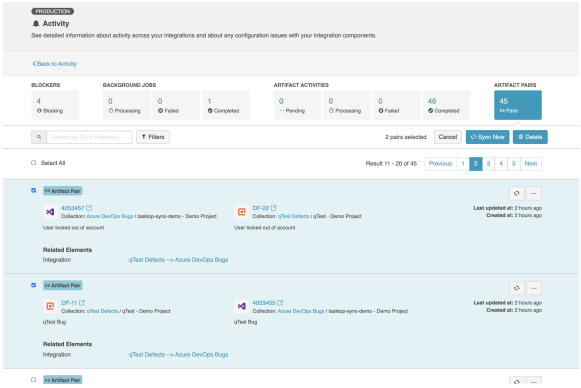
⚠️ **Note**: If too many artifacts are created, this may cause many artifact events to be created — resulting in an influx of pending events that may delay subsequent processing. Please proceed with caution.

Click **Bulk Actions** and then click ✅ **Select ...** to select the artifact pairs you'd like to synchronize or delete.



Once the pairs are selected, the option to synchronize or delete selected pairs will be enabled.

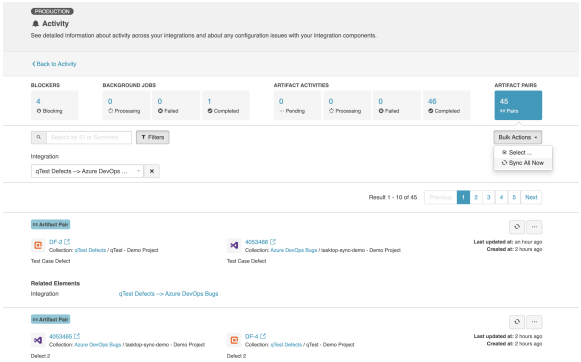
⚠️ **Note**: Conflicts may occur as Tasktop will generate events for both artifacts within the pair. Please refer to the conflict resolution strategy configured within your integrations.



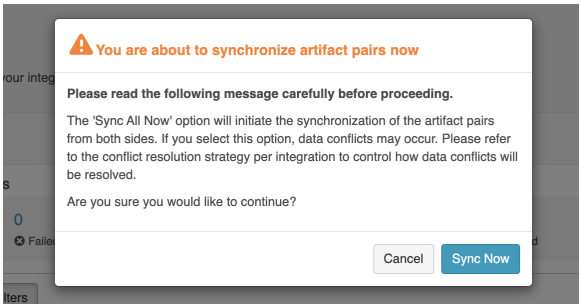
If you'd like to use the bulk option **Sync All Now**, you must first apply a filter to the Artifact Pairs list. This will move all artifact pairs that meet your criteria back to the Artifact Pairs list where you can bulk synchronize them.

To do this, click **Sync All Now**.

💡 **Note**: This option will **only** appear when filters are configured or search text is entered.




A pop-up will appear confirming you'd like to synchronize all selected pairs. Click **Sync Now** to synchronize the artifact pairs.




# Exporting Artifact Pairs

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

You can also migrate artifact pairs from an On-prem instance to a cloud instance using the export functionality.

**Before you Begin**

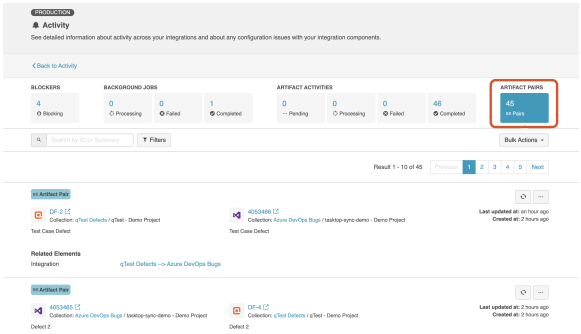
Before using this feature, please ensure that the following requirements are met:

1. To use this feature, you must be enabled for access by Tasktop support.
2. The On-prem instance should be upgraded to the latest (major) version that matches the Cloud instance

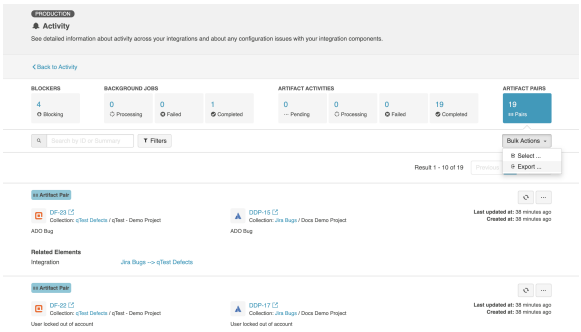
   💡 **Note**: This ensures that the artifact handles from the On-prem instance are compatible with the Cloud instance.

3. The Cloud instance should have the same repositories, integrations, collections, and field mappings configured
4. If using a Keycloak version of Hub, the user logged in must be a **TasktopAdmin** user
5. You **must** be on an On-Premise instance (export **cannot** be enabled **from** a Cloud instance)
6. Before exporting, it is recommended to **stop** all other integrations as you may encounter performance issues that could impact running integrations and the export.

To begin exporting your artifact pairs, navigate to the **Artifact Pairs** tab on the **Activity Screen**.

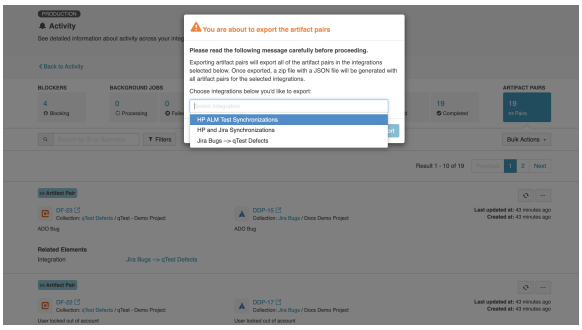



Click the **Export** option under the **Bulk Actions** dropdown.




A pop-up will appear where you can select the integrations from which you'd like to export artifact pairs.

💡 **Note:** Only artifact pairs from work-item synchronization integrations can be exported.



After selecting the integration you'd like to export, click **Export** and a .**zip** file will be generated with a .**json** file containing all the artifact pairs.

**During Export**

Upon export, the selected integration will be disabled and no artifacts will be processed.

If the integration is in the middle of processing artifact events, the export will not start until the artifact events have finished processing. If the event cannot finish processing within the specified time limit, an error will appear alerting you to manually disable the integration before exporting the integration.

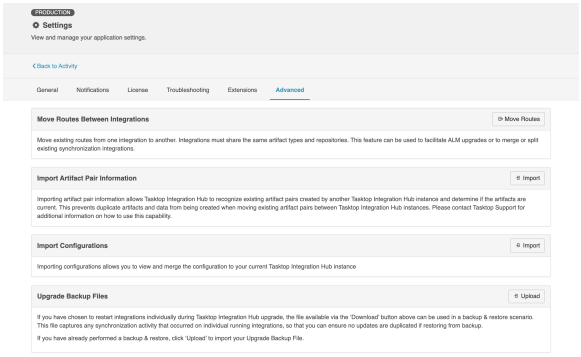
💡 **Note:** Only **one import** and **one export** can occur simultaneously.

If you encounter any issues upon exporting your artifact pairs, please reach out to Tasktop support.

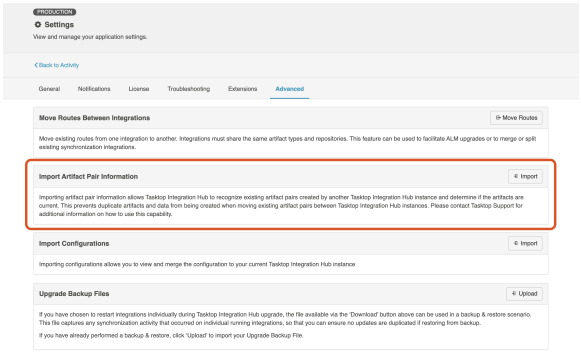
# Importing Artifact Pairs

⚠️ **Note**: If you do not **stop** the exported integration *before* importing into the cloud instance, you may encounter **duplicate** artifacts.

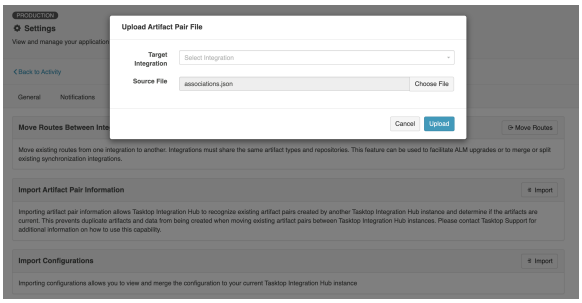
After extracting the exported .**zip** file, you can import the .**json** file in the **Advanced** tab on the **Settings** screen.



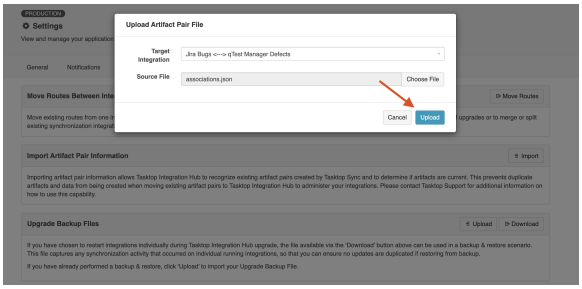

Click **Import** in the **Import Artifact Pair Information** section.




A pop-up will appear where you can select the target integration and the exported .**json** file containing all the artifact pairs.




After you've selected the target integration and source file, click **Upload** to import your artifact pairs.



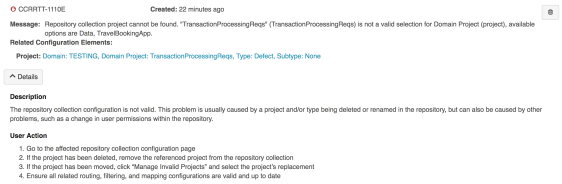

If you encounter any issues upon importing your artifact pairs, please reach out to Tasktop support.

# Specific Error Messages

## Errors on Activity Screen

You can find details on some specific error messages in our FAQ (in the Troubleshooting section) and in our connector pages (for connector-specific errors). We've also outlined errors below which require specific steps in the Tasktop UI.

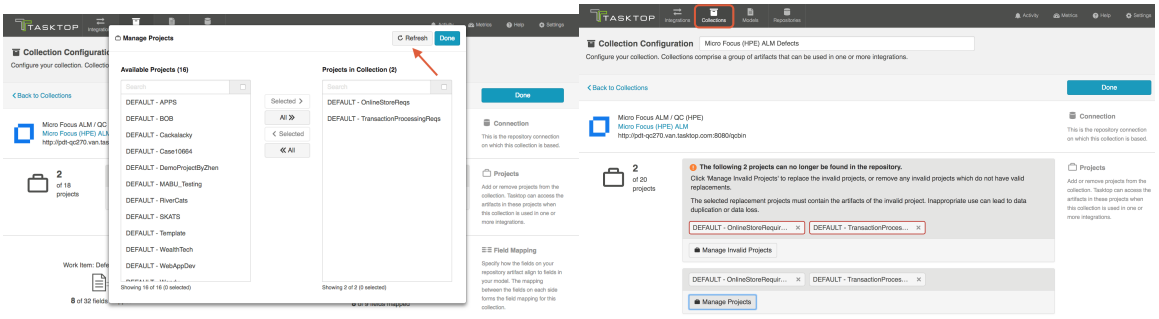
## Repository collection project cannot be found




This error message is usually caused by a project type being deleted or renamed in the repository, but can also be caused by other problems, such as a change in user permissions within the repository, or moving the project to a new domain within that repository.

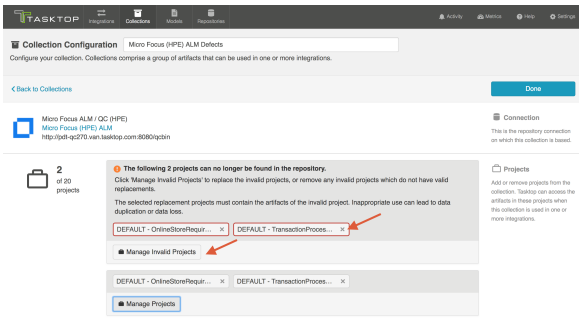
To resolve this error, go to the Collection configuration screen.  Here, you will see a message alerting you to the fact that previously selected project(s) cannot be found in the repository.

💡 Note: You may not see the alert message on the Collections screen until Tasktop's cache refresh occurs.  To 'force' the message to appear, click 'Manage Projects' and then refresh the project schema.  This will cause the alert to appear.

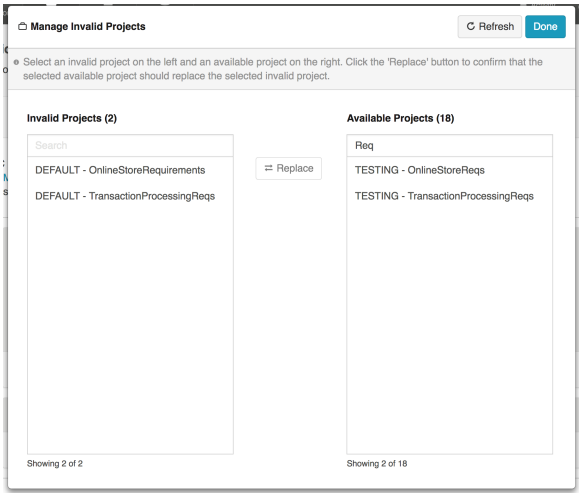



You can click the 'x' to remove any projects which do not have valid replacements, or click the 'Manage Invalid Projects' button to select replacement projects.

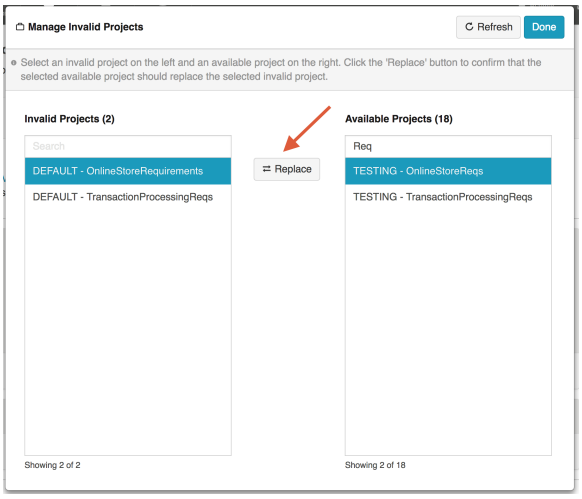
💡 Note: If you remove a invalid project (instead of replacing it via the 'Manage Invalid Projects' button) and then add its replacement to the collection later, you risk creating duplicate artifacts.  Project replacements should always be executed via the 'Manage Invalid Projects' button, and all project replacements should be done at the same time.



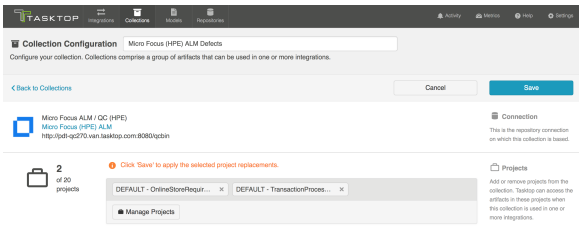

After clicking 'Manage Invalid Projects,' you will see the 'Manage Invalid Projects' picker, where you can search for available project replacements:



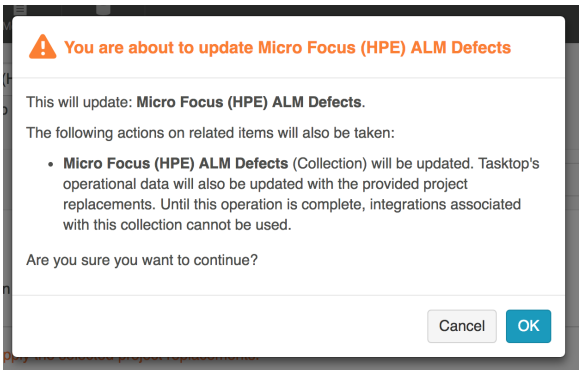

Highlight the invalid project on the left, and its replacement project on the right.  Then click 'Replace.'  Repeat the steps for any invalid projects you'd like to replace, and then click 'Done.'




You will be prompted to save your collection in order to apply the updates (note that until the collection is saved, the invalid project names may display).




You will get a pop-up message warning you that the integrations associated with this collection cannot be used until the project update is complete:



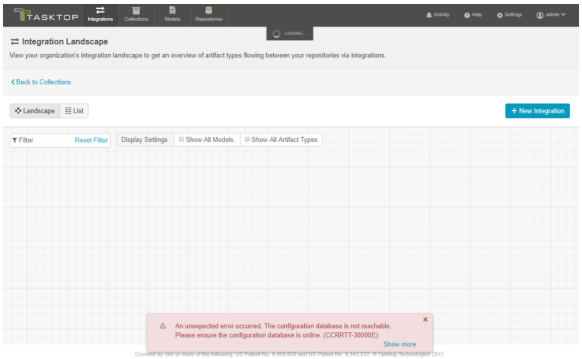

You can view progress for your project replacements on the Background Jobs tab of the Activity screen.

# In-Application Errors

There are some scenarios where you may see an error message within the application itself, rather than on the Activity Screen.

## External Database Error

If you have exported your Tasktop configuration information to an external database (see information here), and your database is not reachable, you will notice that your configuration elements (i.e. repositories, collections, integrations, etc.) will not be visible, and an error message will appear.  To resolve this error, please ensure that your external configuration database is online.



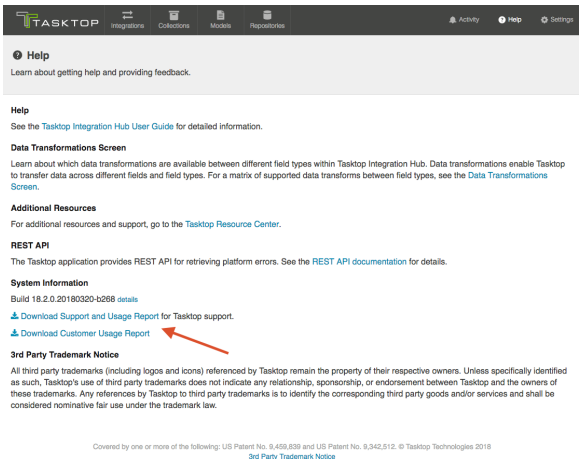
# Support and Usage Reports

## Overview

In cases where the Activity screen is not enough to resolve a problem, a Support and Usage Report is available to provide additional information.

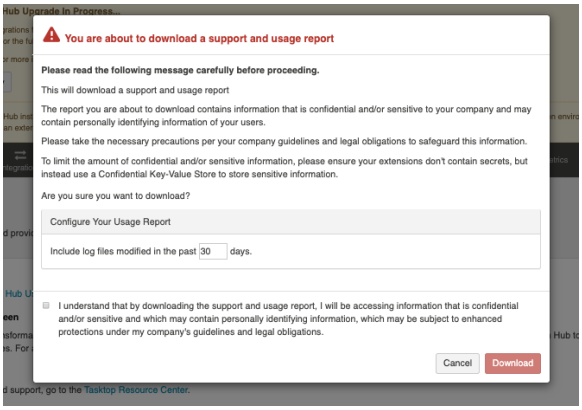
The Support and Usage Report can be downloaded from the **Help** screen.

To download, click the **Download Support and Usage Report** link in the **System Information** section on the Help screen.




Upon downloading, you can specify how many days of log files you'd like to include in the Support and Usage Report.

💡 **Note**: The default value for this field is set to 30 days.




## Report Contents

The downloaded report file is named tasktop-state-DATE-TIME.zip. Once unzipped, there will be five folders. The folders and contents are listed below.

1. activity
    - issues.json
2. configuration
    - configuration.json
    - hub-details.json
3. crash-reports
    - hs_err_pid*.log
4. logs
    - logs by day for past 14 days
    - configuration-changes.log
    - extensions.log
    - thread-dump.log
    - localhost.log
    - localhost_access_log.txt
    - catalina.log
    - tasktop-service.log
    - keycloak-service.log
    - keycloak-stderr.log
    - keycloak-stdout.log
5. mappings
    - text file for each collection configured
6. metrics
    - metrics.json
7. repository metadata
    - file for each repository connection configured
8. schemas
    - JSON file for each collection configured
9. usage
    - usage report
    - overview.json

| Folder | File Name | Contents |
|---|---|---|
| activity | issues.json | Contains issues shown on the Activity screen. |
| configuration | configuration.json | Contains all the configuration of your application instance. |
| configuration | hub-details.json | Contains details about the specific build and license of the application. |
| | | |

| | | |
|---|---|---|
| crash-reports | hs_err_pid*.log | Contains log files generated when the Java Virtual Machine crashes. |
| logs | logs | A separate file is created for every day of logs — 14 days of logs are saved. |
| logs | configuration-changes.log | Contains details on configuration changes made in Tasktop Integration Hub, broken out by user (if applicable) and date/time. Note that the user is identified by their user ID, which can be found in the user administration screen (accessible by Tasktop admins only). |
| logs | extensions.log | Contains any logs generated when an extension is called. The extension will write out a log whenever the console.log function is called. |
| logs | migration-event-trace.log | Contains logs populated only when migrations are running. |
| logs | thread-dump.log | Contains all Tasktop thread information at the point of time the Support and Usage report is downloaded. This file will only be included if your Tasktop instance has crashed or if you have forced Tasktop to close. |
| logs | localhost.log | Tomcat's host log |
| logs | localhost_access_log.txt | Tomcat's log of requests |
| logs | catalina.log | Tomcat's container log |
| logs | tasktop-service.log | Tasktop Windows service log, showing service start and stop |
| logs | keycloak-service.log | Keycloak Windows service log, showing service start and stop |
| logs | keycloak-stderr.log | Keycloak standard error output |
| logs | keycloak-stdout.log | Keycloak standard output |
| mappings | collection-label.txt (i.e. jira-defects.txt) | Contains information about collection mappings with transformation identifiers from Collection to Model and from Model to Collection. |
| | | |

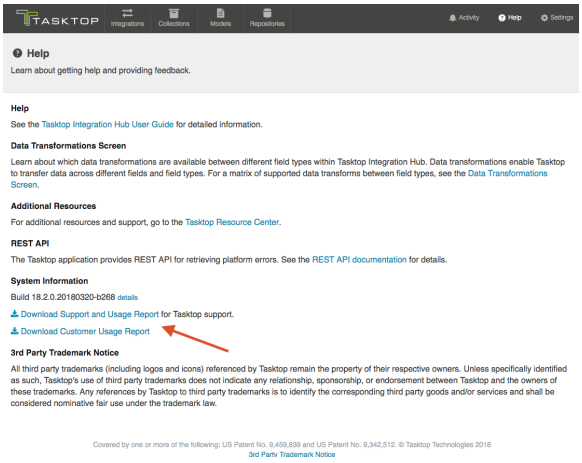
| | | |
|---|---|---|
| metrics | metrics.json | Contains various metrics of the application. |
| metrics | change-detection-metrics.json | Contains metrics relating specifically to integrations and change detection. |
| repository metadata | repository-label.json (i.e. jira.json) | Contains repository metadata (i.e., repository version, repository timezone, repository api rate limit, repository default pagination size, repository additional metadata, connector timezone, repository state) for each repository connection configured. |
| schemas | collection-label.json (i.e. jira-defects.json) | Contains collection schema information (i.e. the same fields that would display on the mapping screen). |
| usage | usage-report.csv | Contains details on Tasktop usage without any with personal information included (i.e., names, e-mail addresses, etc). |
| usage | overview.json | Contains details such as repository versions, number of integrations, integration routes and last processed times, number of activities (creates and updates) by integration and repository, and number of person IDs seen by integration and repository.<br><br>💡 **Note**: Some integrations that do not have artifact associations will not have last processed times. |

## Usage Reports

Tasktop supplies a Usage Report to enable customers to review and understand their Tasktop usage.

Two reports are provided:

- A sanitized report that does not contain personal information (such as names, email addresses, or usernames), that is part of the Support and Usage Report file
- A Customer Usage Report which contains personal information (such as names, email addresses, and usernames), that can be used to analyze and reconcile user counts




Both reports contain the following fields:

- **Tasktop Generated Person Identifier**:
  - This is generated to identify a person that flows between two or more repositories. If Person Reconciliation is in effect, the users that are the same across repositories will have the same Tasktop Generated Person Identifier. This field may be blank in scenarios where a person existed on an artifact seen by Tasktop, but where the field that contained that person did not flow to another repository.
- **Tasktop Generated Repository Person Identifier**:
  - This is generated for each unique person Tasktop sees within one repository. Note that the person field does not need to flow in order to be counted here. Since this is repository-specific, you could see two (or more) different Tasktop Generated Repository Person Identifiers that share the same Tasktop Generated Person Identifier.
- **Connector**:
  - Tasktop's name for the connector
- **Repository Label**:
  - The name (label) supplied by the customer for the repository
- **Integration Name**:
  - The name supplied by the customer for the integration within Tasktop
- **Collection Project**:
  - The collection and project names that contain the person
- **Repository Fields**:
  - The repository fields that the person was seen on during the course of a month
- **Model Fields**:
  - The model fields mapped to the repository fields listed above
- **Count**:
  - The number of times the Tasktop Generated Repository Person Identifier was seen for the given integration/collection/project combo in one month
- **Month**:
  - The month that the count (above) applies to

The customer-facing report also contains the following fields:

- **First Name**

- **Last Name**
- **Display Name**
- **Email**
- **Username**
- **Repository Person ID:**
    - A repository specific identifier.  Some repositories provide an ID that is unique from the username.

💡 **Note**: The customer-specific fields above may be blank depending on the associated repository and whether Tasktop has retrieved them yet (these fields are retrieved periodically).

Both reports contain data collected over a rolling 2 year span.

## Logging Settings

Tasktop provides two logging levels for the logs in the support and usage reports: Normal and Troubleshooting. Please see the Logging section of the Troubleshooting (Settings) screen for more details on how to configure each setting.

# Error Message Appendix

The following is a complete list of error messages. Error messages are displayed on the Activity screen. More details on specific errors can be found under Troubleshooting and in our FAQ.

## CCRRTT-0001E – An unexpected error occurred.

### Description

An unexpected error has occurred.

### User Action

Attempt to resolve error according to the specific error message.

## CCRRTT-0002E – The maximum number of allowable errors has been reached.

### Description

The maximum number of allowable errors has been reached. Any errors encountered after the maximum number will be discarded.

### User Action

1. Open the errors page and resolve the listed errors

## CCRRTT-0003E – The system has run out of memory.

### Description

The system has run out of memory. Services have been stopped.

### User Action

1. Increase the amount of memory available (see help docs).
2. Restart Tasktop Integration Hub.

## CCRRTT-0004E – Configuration migration failed.

## Description

Configuration could not be migrated to match an updated version of Tasktop Integration Hub due to one or more errors.

## User Action

1. Investigate the cause of failure by viewing related errors under Issues on the Activity page.
2. Attempt to resolve error according to the specific error message and corresponding user actions.
3. Retry the configuration migration from the activity page, or restart the Tasktop Integration Hub application.

# CCRRTT-0005E – There is a conflicting artifact association.

## Description

The artifact association could not be imported as an existing artifact association conflicts with it.

## User Action

Contact support for assistance.

# CCRRTT-0006W – Upgrade data migration cancelled.

## Description

Data migration required to run an updated version of Tasktop Integration Hub was cancelled due to a configuration change or because Tasktop Integration Hub was shutdown.

## User Action

None, data migration will be resumed automatically.

# CCRRTT-0006E – Migration cannot be completed as there are errors related to disabled repositories.

## Description

Migration cannot be completed as there are errors related to disabled repositories.

## User Action

1. Open the Activity page and delete all errors related to the specified repository, or
2. Navigate to the Repositories page and enable the specified repository

# CCRRTT-1000E – Unable to communicate with repository.

## Description

There was a network error when attempting to communicate with a repository.

## User Action

1. Check the network connection between Tasktop Integration Hub and the repository.
2. Try connecting again later.

If the problem persists, contact your network administrator.

# CCRRTT-1002E – An unexpected connector error occurred.

## Description

An unexpected connector exception has occurred.

## User Action

Attempt to resolve error according to the specific error message.

# CCRRTT-1003E – An error occurred while executing an operation.

## Description

An exception has occurred during the execution of a connector operation.

## User Action

Attempt to resolve error according to the specific error message.

## CCRRTT-1004E – Connection to LDAP directory failed.

### Description

An unexpected error has occurred while attempting to establish a connection with an LDAP directory.

### User Action

Attempt to resolve error according to the specific error message.

## CCRRTT-1005E – An unexpected error occurred while communicating with an LDAP directory.

### Description

An unexpected error has occurred while communicating with an LDAP directory.

### User Action

Attempt to resolve error according to the specific error message.

## CCRRTT-1104W – Authentication state for repository connection has expired.

### Description

The authentication state for a repository connection has expired.

### User Action

Typically, the authentication state for a repository connection expires on a periodic basis and authentication will be retried automatically. If the error persists, verify that the repository credentials for the associated repository are correct.

## CCRRTT-1105E – The project configuration is invalid.

### Description

The project configuration is not valid. This problem is usually caused by a project and/or type being deleted or renamed in the repository, but can also be caused by other problems, such as a change in user permissions within the repository.

## User Action

1. Determine the cause of the problem from the specific error message
2. Navigate to the referenced repository collection or artifact union
3. Correct the problem on the repository and then click ? *Refresh Projects*?, or
4. Remove the referenced project from the repository collection or artifact union
5. If a project has been renamed add the renamed project back
6. Ensure all related routing, filtering, and mapping configurations are valid and up to date

# CCRRTT-1107E – Connection could not be established with a repository due to a failure during authentication.

## Description

There was an unexpected error while attempting to authenticate with a repository.

## User Action

Attempt to resolve error according to the specific error message.

# CCRRTT-1109E – Project configuration is outdated.

## Description

The project configuration is outdated.

## User Action

1. Identify the outdated project configured from the specific error message
2. Remove the outdated project from the associated Repository Collection or Artifact Union
3. Select ? *Manage Projects*? and press the ? *Refresh*? button
4. Add the project back

# CCRRTT-1110E – Repository collection project cannot be found.

## Description

The repository collection configuration is not valid. This problem is usually caused by a project and/or type being deleted or renamed in the repository, but can also be caused by other problems, such as a change in user permissions within the repository.

## User Action

1. Go to the affected repository collection configuration page
2. If the project has been deleted, remove the referenced project from the repository collection
3. If the project has been moved, click ? *Manage Invalid Projects*? and select the project???s replacement
4. Ensure all related routing, filtering, and mapping configurations are valid and up to date

# CCRRTT-1111E – Repository collection contains duplicate projects.

## Description

The high-level container (i.e. the type of container chosen when clicking ? *Manage Projects*? on the Collections screen) has changed.

## User Action

Before resolving this issue, please:

1. Review and write down the current artifact routing configuration for any integrations utilizing this collection as these must be reconfigured once the issue is resolved.
2. To ensure you understand the changes made to your collection, please navigate to the collection and review what is now selected under ? *Manage Projects.*? No changes will need to be made on this screen.

Once this issue is resolved, your artifact routing will be removed from any relevant integrations, and need to be manually reconfigured.

# CCRRTT-1112E – Artifact is locked.

## Description

The artifact is locked by another user or process.

## User Action

See the specific error message for details on what artifact is locked. Ensure that no other user or process is currently using the artifact, and retry the operation.

## CCRRTT-1113E – Connection could not be established with a repository due to an insecure connection.

### Description

The repository connection could not be established due to an insecure connection.

### User Action

- Attempt to resolve error according to the specific error message, or
- Navigate to the specific Repository and enable the setting for allowing insecure connections

## CCRRTT-1114E – The artifact union configuration is invalid.

### Description

The artifact union configuration is invalid.

### User Action

1. Navigate to the artifact union configuration screen,
2. Correct the invalid configuration according to the specific error message

## CCRRTT-1115I – This message is to notify you that events processing for this repository may be delayed due to the repository???s event rate limit.

### Description

The events processing for this repository may be delayed because the repository???s event rate limit is set too low.

### User Action

1. Navigate to each integration utilizing this repository and update the change detection interval, or

2. Navigate to the repository connection screen and update the event rate limit.

## CCRRTT-1401E – Integration must specify at least one route.

### Description

An integration must contain at least one route.

### User Action

1. Navigate to the integration routing page
2. Add at least one route

## CCRRTT-1402E – Integration must satisfy style constraints.

### Description

An integration must satisfy the constraints of its style. This type of error should not happen when an integration is built using the UI.

See the detailed message for more details about the parts of the integration that are invalid.

### User Action

1. Navigate to the integration page
2. Adjust the configuration to be valid (according to the messages)
3. If this integration was created via the web UI, consider contacting support

## CCRRTT-1403E – Integration must have all collections attached to the same model.

### Description

Collections used in an integration must all be attached to the same model.

### User Action

1. Determine which model the integration should be using
2. Navigate to the integration and determine which collections are not using this model
3. Either remove the identified collections from the integration, or
4. For each identified collection, set the mapping to the correct model

## CCRRTT-1404E – Collection must have a mapping to a model.

### Description

Repository Collections used in an integration must have a mapping to a model.

### User Action

1. Navigate to the collection
2. Select a Model to create a mapping

## CCRRTT-1405E – Integration must have a source Collection.

### Description

An integration must have a source collection.

### User Action

1. Navigate to the Integration
2. Add a collection to be used as a source

## CCRRTT-1406E – Integration must have a target Collection.

### Description

An integration must have a target collection.

### User Action

1. Navigate to the Integration
2. Add a collection to be used as a source

## CCRRTT-1408E – Integration failed to lookup artifact.

### Description

An integration failed to locate the artifact to be modified. This can be caused by:

- a missing formatted ID value on the source artifact,
- an invalid formatted ID value on the source artifact, or
- the absence of a target collection which contains an artifact matched by the formatted ID.

See the detailed message for more details about the parts of the lookup that failed.

## User Action

1. Navigate to the integration page
2. Ensure the key field is configured correctly on the field flow page
3. Ensure the data on the source artifact is correct
4. Ensure a matching artifact is contained in a target collection

# CCRRTT-1409E – Integration has invalid filter.

## Description

The filter used in the integration has become invalid.

## User Action

1. Navigate to the integration filter in error.
2. Resolve each error that appears in the filter.

# CCRRTT-1410E – Integration must specify a key identifier.

## Description

An integration must specify a key identifier for the given collections. Key identifiers are used to determine how to locate artifacts in a target collection. They do this by specifying the field on the source model that contains the target artifact formatted id.

## User Action

1. Navigate to the integration page
2. Select the two collections missing a key identifier
3. Navigate to the field flow page and configure a key identifier

# CCRRTT-1411E – All specified routes of an integration must be configured.

## Description

All specified routes of an integration must be configured.

## User Action

1. Navigate to the integration routing page
2. Configure all routes which require configuration

# CCRRTT-1412E – Integration has a conditional route with invalid configuration.

## Description

The conditional routing configuration of the integration has become invalid.

## User Action

1. Navigate to the integration route in error.
2. Resolve each error that appears in the routing configuration.

# CCRRTT-1413E – Collection has invalid repository query.

## Description

The repository query used in the collection has become invalid.

## User Action

1. Navigate to the collection.
2. Resolve the error by selecting a different repository query.

# CCRRTT-1414I – Tasktop Integration Hub is currently updating its operational data for this integration.

## Description

Tasktop Integration Hub is currently updating its operational data for an integration.

## User Action

1. Wait for the data to be updated.

# CCRRTT-1415E – The routing configuration is invalid.

## Description

The artifact routing for this integration is invalid because a route endpoint no longer exists, or cannot be routed to (i.e. a more specific route endpoint must be selected).

## User Action

1. Navigate to each collection participating in the integration to review what is now selected under 'Manage Projects.'
2. Navigate to each integration and reconfigure the artifact routing. Once the routing is valid, this issue will clear.

# CCRRTT-1416E – The twinless artifact update configuration is invalid.

## Description

The twinless artifact update for this integration is invalid.

## User Action

1. Navigate to the twinless update configuration for this integration.
2. Resolve the error according to the specific error message.

# CCRRTT-10004E – Enterprise Data Stream Integration must have exactly one target SQL Collection.

## Description

An Enterprise Data Stream Integration must reference a single SQL collection.

## User Action

- Select a SQL Collection for the target of the Integration that is in error.

## CCRRTT-10005E – Enterprise Data Stream Integration must have a source Collection.

### Description

An Enterprise Data Stream Integration must reference at least one Collection to be used as a source of artifacts.

### User Action

Select a source Collection for the Integration that is in error.

## CCRRTT-10006E – Enterprise Data Stream Integration target Collection must have appropriate mapping.

### Description

An Enterprise Data Stream Integration???s data Collection must be mapped to a model. This corresponds to the model desired to be reported on.

### User Action

Add mappings for the Collection used in the Enterprise Data Stream Integration.

1. navigate to the Collection
2. add a mapping to a model

## CCRRTT-10007E – Enterprise Data Stream Integration source Collection must provide the correct model.

### Description

An Enterprise Data Stream Integration source Collection must be mapped to the same model as the target Collection.

### User Action

Add relationship to the model for the source Collection used in the Enterprise Data Stream Integration

1. navigate to the Integration
2. identify the model of the target Collection
3. navigate to the source Collection in error, and ensure that its model matches the model of the target Collection
    - if the source collection is a Repository Collection, add a mapping to the corresponding model
    - if the source collection is a Gateway Collection, ensure its model is set to the corresponding model

## CCRRTT-10008E – Enterprise Data Stream Integration target Collection must have exactly one project.

### Description

An Enterprise Data Stream Integration???s Collection must have exactly one project.

### User Action

1. Navigate to the Collection
2. Ensure it has exactly one project which corresponds to the database table

## CCRRTT-10009E – Enterprise Data Stream Integration is missing required column.

### Description

An Enterprise Data Stream SQL Collection???s underlying database table is missing a required column.

### User Action

Add the required column to the underlying database table. See error message for missing column id.

## CCRRTT-15002E – Integration services cannot be started due to a problem with the license.

### Description

Tasktop Integration Hub integration services cannot be started due to a problem with the license. This problem can be caused by running the software without a license, using features that are not included in the installed license, or by having an invalid or expired license.

## User Action

This problem can be resolved by installing a valid license using the following steps:

1. Obtain a valid license by contacting the Tasktop Support Center
2. Navigate to the settings page
3. Press the Apply New License button under License
4. Paste in the license text and press Save

# CCRRTT-15005E – Repository cannot be used due to a problem with the license.

## Description

The repository connection cannot be used because connections to repositories of this type are not enabled by the license.

## User Action

This problem can be resolved by installing a valid license using the following steps:

1. Obtain a valid license by contacting the Tasktop Support Center
2. Navigate to the settings page
3. Press the Edit button under License
4. Paste in the license text and press Save

# CCRRTT-15011E – Your licensed user count has been exceeded.

## Description

Your licensed user count has been exceeded.

## User Action

Please contact your sales representative.

## CCRRTT-15012E – Conditional field flow is not licensed.

### Description

The integration cannot be used because conditional field flow configurations exist in this integration but are not enabled by the license.

### User Action

Perform one of the following:

- Remove the conditional field flow from the integration
- Contact the Tasktop Support Center to obtain and install a license that includes the conditional field flow feature

## CCRRTT-15013E – Artifact unions are not licensed.

### Description

The collection cannot be used because mappings using a field from an artifact union exist within this collection but are not enabled by the license.

### User Action

Perform one of the following:

- Remove the mappings using a field from an artifact union from the collection
- Contact the Tasktop Support Center to obtain and install a license that includes the artifact union feature

## CCRRTT-16001E – Services cannot be started until Tasktop Integration Hub security has been initialized.

### Description

Tasktop Integration Hub integration services cannot be started because secure password storage has not been configured and initialized.

### User Action

1. Navigate to the Settings page

2. Specify the Master Password under Secure Password Storage

## CCRRTT-16002W – The Tasktop Integration Hub services restart is taking longer than expected.

### Description

The Tasktop Integration Hub services restart is taking longer than expected.

### User Action

- Wait for the Tasktop Integration Hub services to restart and this issue will be removed automatically.
- If the Tasktop Integration Hub services do not restart (this issue is still present) after 30 minutes, please contact the Tasktop Support Center for assistance: ? *https://links.tasktop.com/support*?.
- Do not restart Tasktop Integration Hub without assistance from support

## CCRRTT-17001E – Mapping cannot be applied since it is not valid within the current context.

### Description

The mapping cannot be applied since the mapping is not valid for the artifacts in the current context.

### User Action

1. Determine the source of the problem from the specific error message
2. Either update the mapping to match the artifacts and model in use, or
3. Update the corresponding artifact schema to match the mapping, for example by changing a field type

## CCRRTT-17002E – Collection model mapping is invalid.

### Description

The collection model mapping is not valid due to inconsistencies between the collection schema, the model schema and the mapping.

### User Action

1. Determine the cause of the problem from the specific error message
2. Navigate to the mapping
3. Update the mapping to match the collection and model in use, or
4. Update the corresponding collection artifact schema to match the mapping, for example by changing a field type, or
5. Update the model to match the mapping, for example by adding a field, or changing a field type

## CCRRTT-17003E – Artifact could not be created or updated because one or more values cannot be accepted.

### Description

An artifact could not be updated or created because one or more of its values are not valid. See the specific error message for details.

### User Action

1. Identify the fields and values that are in error from the specific error message
2. Correct the source data, either by
   - updating the source artifact, or
   - by making changes to the mapping, or
   - by making changes to the target system so that the provided data is valid, or
   - by providing a new artifact via a Gateway Collection

## CCRRTT-17004W – Artifact cannot be processed since it is currently in use.

### Description

Artifact cannot be processed since it is currently in use. This temporary problem occurs when Tasktop Integration Hub attempts to process changes to an artifact concurrently.

### User Action

This error will resolve itself automatically, no user action required.

## CCRRTT-17005E – Field flow is invalid.

### Description

The field flow configuration is not valid due to inconsistencies between the the model schema and the field flow.

## User Action

1. Determine the cause of the problem from the specific error message
2. Navigate to the integration
3. Select the collection pair
4. Navigate to the field flow
5. Update the field flow to match the model in use, or
6. Update the model to match the field flow, for example by adding a field

# CCRRTT-17006E – Artifact was created but some values could not be set.

## Description

An artifact was created by an integration but some values on the artifact could not be set.
The resulting artifact has some field values that may not be correct.

## User Action

1. Determine the cause from the specific error message
2. Either retry the corresponding activity, or
3. Verify the state of the created artifact and manually adjust values as necessary

# CCRRTT-17007E – Conflict resolution strategy is invalid.

## Description

The conflict resolution strategy configuration is invalid.

## User Action

1. From the integration, navigate to the conflict resolution strategy
2. Select an option for the conflict resolution strategy

# CCRRTT-17008E – Artifact could not be processed as it did not meet any of the configured conditions on the Conditional Artifact Routing page.

## Description

Artifact could not be processed as it did not meet any of the configured conditions on the Conditional Artifact Routing page.

## User Action

- Update the conditions configured on the Conditional Artifact Routing page to ensure the artifact???s field value is accounted for, or
- Update fields on the artifact to ensure that it meets the conditions set on the Conditional Artifact Routing page, or
- Update specification for handling artifacts not matched by conditions configured on the Conditional Artifact Routing page to ? *Ignore*? or ? *Default Route*? instead of ? *Error*?.

# CCRRTT-17009E – Invalid state transition.

## Description

An extension provided invalid values when attempting to transition an artifact.

## User Action

1. Identify the extension that produced invalid values
2. Identify the fields and values that are in error from the specific error message
3. Modify the extension to produce a valid transition

# CCRRTT-17010E – Repeated state transition.

## Description

An extension attempted to transition an artifact with the same transition more than once.

## User Action

1. Identify the extension from the error message
2. Modify the extension to avoid repeated transitions of the same type for an artifact

# CCRRTT-17011E – Extension completed with an error.

## Description

An extension completed with an error. See the specific error message for details.

Extensions complete with errors for one of two reasons:

- the extension intentionally raised an error, for example to indicate that a business rule was not satisfied
- the extension itself has an error in its implementation

## User Action

1. Determine from the specific error message the cause of the error
2. Either modify the extension to prevent the error from occurring, or
3. Modify the source or target artifact to satisfy the condition that caused the error

# CCRRTT-17013E – The state transition requires the selection of model fields.

## Description

A state transition extension is configured in a collection that has no model fields selected.

## User Action

Either disable the state transition of the collection or select model fields for the state transition.

To select the fields for the state transition:

1. navigate to the collection
2. navigate to the collection state transitions via the ? *Configure State Transition*? link
3. add the model fields required by the state transition in "State Transition Fields"

To disable state transitions in the collection:

1. navigate to the collection
2. navigate to the collection state transitions via the ? *Configure State Transition*? link
3. select ? *None*? for "State Transition"

# CCRRTT-17014E – Relationship values could not be resolved during synchronization.

## Description

One or more relationship links could not be resolved as part of a synchronization.

This problem occurs when two artifacts that link to each other are synchronized out of order.
This commonly occurs when one artifact (A) links to another (B), but the linked-to artifact B has not yet been synchronized.

When the copy of artifact A (A') is created in the target repository, a link to a copy of B (B') cannot be created at that time since B' has not yet been created.
This problem usually resolves itself once B' is created; the link from A' to B' is created once B' becomes available.

## User Action

- None; wait for the error to be resolved automatically, or
- Remove the unresolved link from the artifact being synchronized

# CCRRTT-17015E – Relationship values could not be resolved during synchronization.

## Description

One or more relationship links could not be resolved as part of a synchronization.

This commonly occurs when one artifact (A) links to another (B), but the linked-to artifact B has more than one corresponding copy in the target repository. This can be caused by having separate synchronization integrations that cause B to be copied into the target repository.

## User Action

- Remove the link from A to B, or
- Use the Artifact Pairs tab on the Activity page to delete any invalid or outdated associations involving B, or
- Remove any unnecessary synchronization integrations which can affect B

# CCRRTT-17016E – An unexpected error occurred when creating the artifact.

## Description

An unexpected error occurred when creating the artifact. The artifact may or may not have been created.

## User Action

1. Do not retry the event without guidance from Tasktop Support,
2. Contact the Tasktop Support Center for assistance: "https://links.tasktop.com/support"

# CCRRTT-17017E – The repository does not support artifact creation.

## Description

The repository does not support artifact creation.

## User Action

1. Navigate to the corresponding integration,
2. Disable artifact creation flow into the specified collection,
3. Remove all routes flowing into the specified collection.

# CCRRTT-17018E – Model does not have all fields required by the state transition.

## Description

A state transition extension is configured in a collection that requires fields that are not configured in the model.

## User Action

Either remove the missing fields in the state transition configuration, or ensure that the model has the required fields.

To add the fields to the model:

1. navigate to the model
2. add the fields

To change the required fields of the state transition extension from the collection:

1. navigate to the collection
2. navigate to the collection state transitions via the ? *Edit state transition*? link
3. modify the list of model fields

## CCRRTT-17019E – Target collection partition could not be resolved during synchronization.

### Description

The work item artifact could not be synchronized due to a missing or invalid route.

### User Action

1. Verify which container this artifact is in in the repository, and ensure that either that container or one of its ancestors has been configured as part of a mirrored container structure; or
2. Ensure that a route has been created for the container in which this artifact originates in the work item integration; or
3. Ensure that the target container has not been deleted. If it has, and if an error exists for it, re-create the container on the Errors screen. To ensure you see an error for the deleted container, make a change to the still-existing corresponding container in the other collection.

## CCRRTT-17020E – Associated target container could not be resolved during synchronization.

### Description

The artifact could not be synchronized because the target container could not be found.

### User Action

1. No action needed, the synchronization should be fixed automatically when the containers synchronize.

## CCRRTT-17021E – An error occurred when performing state transitions.

### Description

A transition was attempted on an artifact but an error resulted.
The artifact may be in an incorrect state.

### User Action

Either address the cause from the specific error message, or disable/reconfigure the state transition of the collection.

1. If the specific error message has a cause, verify the state of the target artifact and manually adjust values as necessary

To disable/reconfigure state transitions in the collection:

1. navigate to the collection
2. navigate to the collection state transitions via the ? *Configure state transition*? link
3. adjust the relevant state transitions

## CCRRTT-17022E – The associated container could not be found.

### Description

The container associated with the parent container of this artifact could not be found.

### User Action

- If the parent container is configured in a route, update the routing configuration to use an existing container
- If the parent container is synchronized by an integration, update the parent container to generate an event for the parent container, and use the ? *Recreate Artifact*? action

## CCRRTT-17023E – For the artifact pair import to succeed, the associated integration must be running.

### Description

For the artifact pair import to succeed, the associated integration must be running.

### User Action

- Run the integration associated with the artifact pair file.

## CCRRTT-17024E – An error occurred when processing the output of an extension.

## Description

An error occurred when processing the output of an extension. See the specific error message for details.

## User Action

1. Determine from the specific error message the cause of the error
2. Either modify the extension to prevent the error from occurring, or
3. Modify the source or target artifact to satisfy the condition that caused the error

# CCRRTT-17025E – Field value configuration required.

## Description

Field value configuration required. The field type for a field in your integration has changed and its field values must be re-mapped.

## User Action

1. Navigate to the appropriate collection (linked above)
2. Go to Field Mapping screen
3. Click ? *Configure*? next to any fields with a ? *!*? icon
4. Map the field values and save
5. Important: Remove this issue to re-enable the integration

# CCRRTT-17026E – Comment flow is invalid.

## Description

The comment flow configuration is not valid. This can happen if a repository???s comment visibility support has changed.

## User Action

1. From the integration, navigate to the comment flow
2. Configure the comment flow as desired

# CCRRTT-17027E – The conditional field flow has an invalid condition.

## Description

The Conditional Field Flow configuration is not valid. This can happen if either a field or value is not resolvable in the model schema.

## User Action

1. From the integration, navigate to the Field Flow
2. Configure the Conditional Field Flow so that all fields and values configured in the conditions are in the respective model used.

# CCRRTT-17028E – Test step model mapping is invalid.

## Description

The test step model mapping is not valid due to inconsistencies between the test step schema, the test step model schema and the mapping.

## User Action

1. Determine the cause of the problem from the specific error message
2. Navigate to the test step mapping
3. Update the mapping to match the collection and model in use, or
4. Update the corresponding test step schema to match the mapping, for example by changing a field type, or
5. Update the test step model to match the mapping, for example by adding a field, or changing a field type

# CCRRTT-17029E – Time entry model mapping is invalid.

## Description

The time entry model mapping is not valid due to inconsistencies between the time entry schema, the time entry model schema and the mapping.

## User Action

1. Determine the cause of the problem from the specific error message
2. Navigate to the time entry mapping
3. Update the mapping to match the collection and model in use, or
4. Update the corresponding time entry schema to match the mapping, for example by changing a field type, or

5. Update the time entry model to match the mapping, for example by adding a field, or changing a field type

## CCRRTT-17030W – Configuration delta table is currently unavailable as it is being migrated.

### Description

Configuration delta table is unavailable as it is being migrated. This temporary problem may occur when a user attempts to access the list of configuration changes before the migration completes shortly after an upgrade.

### User Action

The configuration delta table migration will complete as a background job, no user action required.

## CCRRTT-17031E – Failed to transform imported configuration change.

### Description

Failed to transform imported configuration change into change applicable for this instance due to missing elements.

### User Action

Create the required missing elements before re-importing the configuration change.

## CCRRTT-20000E – No integration is listening to the Gateway Collection.

### Description

A Gateway Collection has been used, but the collection is not configured as a source in an integration. The payload has been lost.

### User Action

1. Use the Gateway Collection in an integration, or

2. Stop pushing to the collection (from the external source)

## CCRRTT-20004E – Relationship fields of a Gateway Collection must be configured to specify the related repository.

### Description

A Gateway Collection must configure the Relationship(s) fields to associate them with the repository having referenced artifacts.

### User Action

1. Navigate to the Gateway collection
2. Locate the ? *Relationship Field Configuration*? section in the UI
3. For each field, select the repository that is associated with that relationship.

## CCRRTT-20005E – Gateway collection must have a model.

### Description

A Gateway Collection must have a model configured.

### User Action

1. Navigate to the Gateway collection
2. Select a model and save the changes

## CCRRTT-20006E – Gateway Collection cannot be used with the configured payload transformation extension due to a restriction in the license.

### Description

A gateway collection has been configured with a payload transformation extension, which is not permitted by the current license.

### User Action

Perform one of the following:

- Delete the offending gateway collection
- Remove the payload transformation extension from the offending gateway collection

## CCRRTT-20007E – Gateway collection must use a token.

### Description

A Gateway Collection must use a token.

### User Action

1. Navigate to the Gateway collection
2. Generate a token and save the changes

## CCRRTT-21001E – An unexpected error occurred while sending an email.

### Description

An error occurred while attempting to send an email.

### User Action

1. Verify that the email settings are specified correctly in the settings
2. Attempt to resolve error according to the specific error message

## CCRRTT-21002E – Failed to authenticate with the email server.

### Description

The mail server rejected the client connection because it was not able to authenticate.

### User Action

1. Verify that the email settings are specified correctly in the settings
    - Double-check the email server hostname and port
    - Double-check the email server credentials
2. Attempt to resolve error according to the specific error message

## CCRRTT-22001E – Artifact Association records with unknown Artifact Handles found and deleted during upgrade.

### Description

During database upgrade, one or more associations were discovered to have an invalid reference. The records that reference nonexistent associated records were logged and deleted.

### User Action

Do not cancel this error or run the associated integration without consulting Tasktop Support. (https://links.tasktop.com/support)

## CCRRTT-30000E – An unexpected error occurred.

### Description

An unexpected error has occurred. Check the specific error message for details.

### User Action

Check the specific error message for details of the failure. If possible correct the problem described in the error message, or contact your administrator for assistance.

## CCRRTT-30001E – Not found.

### Description

The entity was not found because the entity no longer exists on the server.

### User Action

Ensure that the provided entity id is correct, and if not correct the id and try again.

## CCRRTT-30002E – The data provided was not valid.

### Description

The data provided was not valid. See the specific error message for details.

## User Action

Correct the problem described in the specific error message and try again.

# CCRRTT-30003E – The connector kind was not found.

## Description

The connector kind was not found.

## User Action

Ensure that the connector kind is specified correctly and try again.

# CCRRTT-30004E – The request entity was not valid JSON.

## Description

The request entity was not valid JSON.

## User Action

Ensure that the request payload is formatted as a valid JSON entity and try again.

# CCRRTT-30005E – Secure password storage must be initialized.

## Description

Secure password storage has not been initialized.

## User Action

Configure secure password storage via the settings page.

# CCRRTT-30006E – Error communicating with {0} repository.

## Description

Error connecting to repository. See the specific error message for details.

## User Action

Check the specific error message for details of the failure. If possible correct the problem described in the error message, or contact your administrator for assistance.

# CCRRTT-30007E – Error processing request MIME attachment.

## Description

The request MIME attachment could not be accepted either due to a bad request or an I/O failure.

This problem can be caused by insufficient disk space or lack of write permissions in the Tasktop Integration Hub application temporary directory.

## User Action

1. Verify that the temporary directory of the Tasktop Integration Hub application is writable,
    - The Tasktop Integration Hub application must have write permissions to the directory
    - The directory must have sufficient available space
2. Try again

# CCRRTT-30008E – Tasktop Integration Hub is stopped, see the Activity View and error log for more details.

## Description

Tasktop Integration Hub has been stopped due to unrecoverable errors. See error log for more details.

## User Action

Correct the problem described in the specific error message and restart.

# CCRRTT-30009E – The database is not available.

## Description

The configuration database is unavailable.

## User Action

Ensure the configuration database is online and can be reached and ensure Tasktop Integration Hub???s database settings are correct.

# CCRRTT-30010E – Connection settings are not valid.

## Description

The provided connection settings are not valid. See the specific error message for details.

## User Action

Correct the problem described in the specific error message and try again.

# CCRRTT-30011E – The database is locked for maintenance and cannot currently be used.

## Description

The configuration database is locked for maintenance and cannot be used.

## User Action

Wait for the ongoing maintenance to complete.

# CCRRTT-30012E – The database is in use by another instance of the application.

## Description

The Configuration database is in use by another instance of the application.

## User Action

If this is the Tasktop Integration Hub instance which should be running, then shut down any other instances of Tasktop Integration Hub using the same database and restart this instance. Otherwise shut down this instance of Tasktop Integration Hub.

## CCRRTT-30013E – Temporary error communicating with {0} repository.

### Description

Temporary error connecting to repository. See the specific error message for details.

### User Action

Retry your action. If the problem persists, contact your administrator for assistance.

## CCRRTT-30014E – Error communicating with repository. Insecure connections are not allowed.

### Description

The repository connection could not be established due to an insecure connection.

### User Action

- Attempt to resolve error according to the specific error message, or
- Navigate to the specific Repository and enable the setting for allowing insecure connections

## CCRRTT-30015E – Deployment configuration error.

### Description

Configuration applicable to the current deployment is incomplete or invalid.

### User Action

Contact Tasktop customer support.

## CCRRTT-30016E – Unauthorized user error.

### Description

The current user is not authorized due to a restriction in the license.

## User Action

Contact your Tasktop Integration Hub administrator.

# CCRRTT-30017E – The license is expired.

## Description

Tasktop integration services cannot be started because the current license has expired.

# CCRRTT-30018E – No license has been configured.

## Description

Tasktop integration services cannot be started because the no license has been configured.

# CCRRTT-30019E – The application is currently starting up.

## Description

The application is starting up and cannot be accessed at this time.

## User Action

Wait for the application to finish starting up

# CCRRTT-30020E – An error occurred when reading the database connection settings.

## Description

The connection settings for the operational database are inaccessible.

## User Action

Ensure the tasktop-db.json file is correctly formatted and the Tasktop Integration Hub user has permission to read it.

## CCRRTT-30021E – The history page is not available before the necessary data migrations are complete.

### Description

The history page is not available before the necessary data migrations are complete.

### User Action

Wait for the migrations to complete.

## CCRRTT-30022W – Repository Connection disabled.

### Description

The repository connection will be disabled until it is manually enabled.

### User Action

- Enable the repository connection manually on the Repository Connection page, or
- Leave the repository connection disabled and ignore this warning

## CCRRTT-50001E – Unable to propagate artifact changes since the target artifact has been removed.

### Description

Changes to an artifact cannot be propagated to the corresponding artifact in the alternate repository of a synchronization integration since the target artifact has been removed.

### User Action

- Use the ? *Recreate Artifact*? action to have Tasktop Integration Hub recreate the artifact that was deleted in the end system and associate it with the still-existing artifact in the other repository (putting them in sync with one another), or
- Delete the associated artifact, or
- Move the associated artifact out of its collection such that the artifact is no longer synchronized, or

- Apply an artifact filter to ensure updates to the artifact will not be synchronized. To do so, make sure the artifact does not meet the filter criteria specified and make sure to configure the filter to apply to artifact updates

## CCRRTT-50002E – A conflict has occurred during synchronization.

### Description

A field conflict was detected when synchronizing artifacts. A field conflict occurs when the value of a field that is set to flow bidirectionally conflicts across your repositories.

The synchronization of these artifacts was halted with an error because a conflict resolution strategy of ? *Error Upon Conflict*? was configured and the system was unable to propagate
the value from either artifact without overwriting a change from the other artifact.

### User Action

- Change the conflict resolution strategy to have one of the repositories dominate in case of a conflict, or
- Manually change the conflicting value on at least one of the artifacts such that there is no longer a conflict, or
- Change the field flow of the affected field to be unidirectional (in which case a conflict is not possible)

## CCRRTT-50005E – A conflict has occurred during synchronization.

### Description

A conflict was detected when synchronizing artifact containment. A conflict occurs when one or more containers of synchronized artifacts is changed for both artifacts.

### User Action

- Change the container of one or both artifacts to its original value or
- Change the conflict resolution strategy to have one of the repositories dominate

## CCRRTT-50006E – Unable to update artifact due to values for dependent single selects not found.

## Description

Unable to find a new value for an unchanged dependent field.

## User Action

- From the error message find the field that the field in error depends on
- In the repository add a value with the same label as the one provided in the error message

OR

- Change the field that the field in error depends on back to its original value

OR

- Remove the mapping for the field that the field in error depends on

# CCRRTT-50007E – Multiple matching containers were found.

## Description

Multiple matching containers were found when attempting to match containers.

## User Action

- Disable container matching in the container mirroring configuration, or
- Rename the containers such that only one container matches, or
- Change the container matching configuration to choose the first matching container, or
- Change the container matching configuration to match containers differently

# CCRRTT-50008E – This integration cannot be started because a required relationship cannot be resolved.

## Description

The integration cannot be started because a required Relationship field cannot be resolved.

## User Action

- Create an integration to synchronize the artifacts referenced by the specified field, or
- Add a constant mapping to the specified field.

## CCRRTT-50009E – Time Tracking integration model must have a field of type Time Entries.

### Description

Model used in a Time Tracking integration must have a field of type Time Entries.

### User Action

Either

1. Navigate to the model
2. Add a field of type Time Entries

Or

1. Create or select another model having a field of type Time Entries
2. Ensure that each collection used in the integration is using the selected model

## CCRRTT-50010E – Time Tracking integration Collection must have a field mapping to a field of type Time Entries in the Model.

### Description

Collections used in a Time Tracking integration must have a field mapped to the model Time Entries field.

### User Action

1. Navigate to the collection model mapping
2. Add a field mapping to the model Time Entries field

## CCRRTT-50011W – Time Tracking integration target Collection does not support impersonation of the Worker field.

### Description

The selected collection does not support worklog impersonation and so has limited use as the target in a Time Tracking integration.
The worklogs will be filed under the user of the target repository connection.

## CCRRTT-50012E – Time Tracking integration Collection does not support time entry filtering.

### Description

Time entry filtering is configured for a collection pair but the source collection does not support it.

### User Action

1. Navigate to the integration
2. Select the collection pair
3. Navigate to Time Entry Filtering
4. Disable the filter

## CCRRTT-50013W – Artifact cannot be created currently as other artifact creations are being processed.

### Description

Artifact cannot be created currently as other potentially conflicting artifact creations are being processed. This temporary problem can occur when Tasktop Integration Hub attempts to create artifacts on both sides of an integration concurrently.

### User Action

This error will resolve itself automatically, no user action required.

## CCRRTT-50014E – The test step flow configuration is invalid.

### Description

The test step flow configuration is invalid.

### User Action

1. Navigate to the test step flow configuration screen,
2. Correct the invalid flow configuration according to the specific error message

## CCRRTT-50015E – The routing configuration for this container + work item integration is invalid.

### Description

The artifact routing for this container + work item integration is invalid because one side of the integration has multiple container collections of the same type, and artifacts are flowing away from that side.

### User Action

1. Navigate to each integration and reconfigure the artifact routing. Once the routing is valid, this issue will clear.
2. When multiple container collections of the same type exist, integrations can only routed toward that side of the integration.
3. Ensure artifact flow is not bidirectional.

## CCRRTT-50016E – Artifact could not be processed as it did not meet any of the configured conditions on the Shared Container Mirroring page.

### Description

Artifact could not be processed as it did not meet any of the configured conditions on the Shared Container Mirroring page.

### User Action

- Update the conditions configured on the Shared Container Mirroring page to ensure the artifact???s field value is accounted for, or
- Update fields on the artifact to ensure that it meets the conditions set on the Shared Container Mirroring page, or
- Update specification for handling artifacts not matched by conditions configured on the Shared Container Mirroring page to ? *Ignore*? or ? *Default Type*? instead of ? *Error*?.

## CCRRTT-50017E – The integration is missing required field mappings.

### Description

Container + Work Item integrations using Shared Container Mirroring conditions require the model field within the condition to be mapped within its associated collections.

### User Action

1. Navigate to the associated collections,
2. Add a field mapping to and from the model field used within the Shared Container Mirroring conditions

## CCRRTT-50018E – The artifact could not be processed as the artifact it depends on has not synchronized.

### Description

The artifact could not be processed as the artifact it depends on has not synchronized.

### User Action

- Ensure that the artifact specified within the detailed error message is included in the integration,
- Wait for it to synchronize

## CCRRTT-50019E – Container matching cannot be enabled when the parent field is mapped with an extension.

### Description

Container matching cannot be enabled when the parent field is mapped with an extension.

### User Action

- Remove the extension from the parent field mapping within the related collections, or
- Disable container matching in the related integration

## CCRRTT-60001E – Error initializing password encryption.

### Description

Secure password storage requires 256-bit AES encryption which is not available in the Java runtime environment.

### User Action

This problem can be resolved by installing the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files in the Java runtime environment. The download is available from oracle.com incl uding a README file with installation instructions.

Alternatively, the unencrypted level of the password store maybe used.

## CCRRTT-61001E – Connector is missing requirements.

### Description

The connector requirements are not met.

### User Action

Read the connector-specific error message to determine which requirements are unsatisfied.

To provide 3rd party components such as a library or SDK, follow the following steps:

1. Navigate to the ? *Repositories*? screen.
2. Select the repository for which the requirements were unsatisfied.
3. On the repository connection screen, provide the required files.

## CCRRTT-61101E – Connection credentials were not accepted by the repository.

### Description

There was an authentication error while attempting to communicate with a repository.

### User Action

1. Verify that the credentials for the associated repository are correct in the settings.

If these steps do not resolve the error, ensure that the user has sufficient permissions in the target repository to create and edit artifacts.

## CCRRTT-61102E – Connection HTTP proxy credentials were not accepted by the repository.

### Description

There was an authentication error with the proxy server while attempting to communicate with a repository.

### User Action

1. Verify that the proxy credentials for the associated repository are correct in the settings.

If these steps do not resolve the error, contact your network administrator for assistance.

## CCRRTT-61103E – Connection settings are invalid.

### Description

The connection settings are invalid.

### User Action

1. Open the connection settings page for the repository that is in error.
2. Update the connection???s settings to valid values.

If these steps do not resolve the error, contact support for additional assistance.

## CCRRTT-61104E – Tasktop Integration Hub is unable to communicate with this repository as it is experiencing high server load.

### Description

Tasktop Integration Hub is unable to communicate with this repository as it is experiencing high server load. This problem is usually caused by exceeding the number of API calls a repository can receive or otherwise placing a high load on the repository.

## User Action

This error will resolve itself automatically when the repository is no longer experiencing high server load. You can also set an event rate limit on the repository connection screen in Tasktop Integration Hub to limit the number of Tasktop Integration Hub events processed for this repository per minute.

# CCRRTT-61105I – This message is to notify you that Tasktop Integration Hub had suspended communication with a repository due to high server load on that repository. Communication has since resumed.

## Description

This message is to notify you that Tasktop Integration Hub had suspended communication with a repository due to high server load on that repository. Communication has since resumed.

## User Action

None.

# CCRRTT-61106E – Repository Connection disabled.

## Description

The repository connection will be disabled until it is manually enabled.

## User Action

- Enable the repository connection manually on the Repository Connection page, or
- Leave the repository connection disabled and ignore this warning

# CCRRTT-63001E – Integration services cannot be started since the current license has expired.

## Description

Tasktop Integration Hub integration services cannot be started because the current license has expired.

## User Action

This problem can be resolved by installing a valid license using the following steps:

1. Obtain a valid license by contacting the Tasktop Support Center
2. Navigate to the settings page
3. Press the Apply New License button under License
4. Paste in the license text and press Save

# CCRRTT-64001E – Integration cannot be used with the configured repositories due to a restriction in the license.

## Description

An integration cannot be run because it is configured with repository pairs which are invalid under the current license restrictions.

## User Action

Perform one of the following:

- Delete the offending integration
- Disable the offending integration
- Update the offending integration to use repository pairs allowed under the current license restrictions

# CCRRTT-65001E – Extension cannot be used because of a restriction in the license.

## Description

A value transformation extension is present which is not permitted by the current license.

## User Action

Perform one of the following:

- Provide a license that includes extensions of this type, or
- Remove extension by navigating to the the Settings -> Extensions page

## CCRRTT-66001I – Tasktop Integration Hub is currently updating its operational data with a collection’s project replacements.

### Description

Tasktop Integration Hub is currently updating its operational data with a collection???s project replacements.

### User Action

1. Wait for collection update to complete.

## CCRRTT-66002I – Tasktop Integration Hub is currently updating its operational data with a collection’s project replacements.

### Description

Tasktop Integration Hub is currently updating its operational data with a collection???s project replacements.

### User Action

1. Wait for collection update to complete.

## CCRRTT-66003W – Integration data migration is currently in progress.

### Description

A background job is currently in progress.

### User Action

Wait for the background job to complete.

## CCRRTT-67000E – Target repository must contain valid containers for this integration.

### Description

Could not find a container where the target artifact could be created.

### User Action

1. Identify the target repository where the container could not be located for the target artifact from the specific error message.
2. Ensure a container that satisfies the following constraints is created in the target repository:
    - Is able to contain the target artifact.
    - Is a sibling of the container of the referenced artifact.

## CCRRTT-67001E – A field mapping must exist from a model String or Rich Text field to a target Relationship or Relationships field.

### Description

A field mapping must exist from a String or Rich Text field in the model to a Relationship or Relationships field in the target collection. Such a mapping is required to determine where to flow any commit artifacts processed by this integration.

### User Action

1. Configure a field mapping from your model Commit Message field to a Relationship or Relationships field in the target collection.

# Metrics

*See Tasktop Editions table to determine if your edition contains basic or advanced Metrics functionality.*

## Introduction

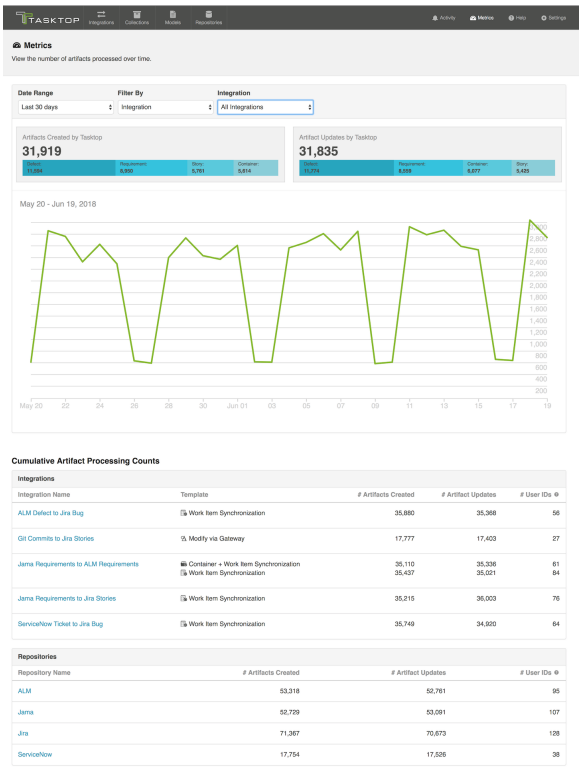
Tasktop Integration Hub provides a **Metrics dashboard** to help you better understand Tasktop activity such as:

- Number of artifacts created by Tasktop
- Number of artifact updates by Tasktop

These metrics are a great tool to:

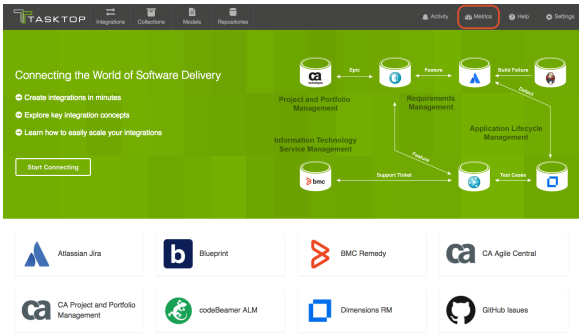
- Understand and troubleshoot downtime
- Communicate the value of Tasktop to your organization
- Analyze trends and patterns within your organization, such as:
  - Are there certain times of year when higher quantities of customer requests flow from your CRM tool to your Requirements tool?
  - Have defects flowing from your ITSM tool to your Agile tool decreased over time?
  - ...and more!

The data used to create the metrics refreshes each time the page is reloaded.



# Viewing the Metrics Dashboard

To access the Metrics Dashboard, click **Metrics** in the right corner of the screen.



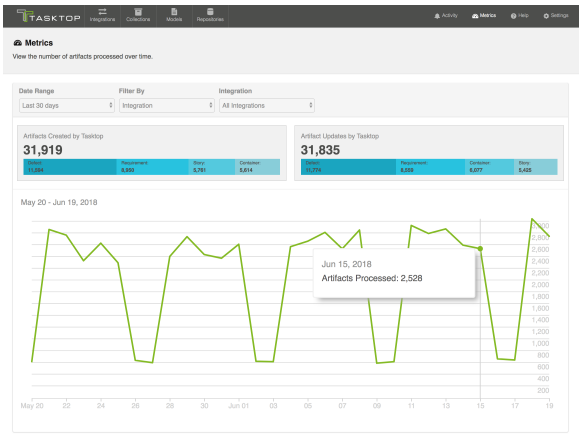

## Basic Functionality

*See Tasktop Editions table to determine if your edition contains basic or advanced Metrics functionality.*

Users with basic functionality will be able to view metrics showing the following:

- Total Artifacts Created
- Total Artifact Updates
- To help understand which artifact types are being synchronized, a blue bar will show the distribution for the metrics above based on model

Metrics above are displayed to show data for all integrations, over the last 30 days.




## Advanced Functionality

*See Tasktop Editions table to determine if your edition contains basic or advanced Metrics functionality.*

Users with advanced functionality will be able to view metrics showing the following:

- Total Artifacts Created

- Total Artifact Updates
- To help understand which artifact types are being synchronized, a blue bar will show the distribution for the metrics above based on model

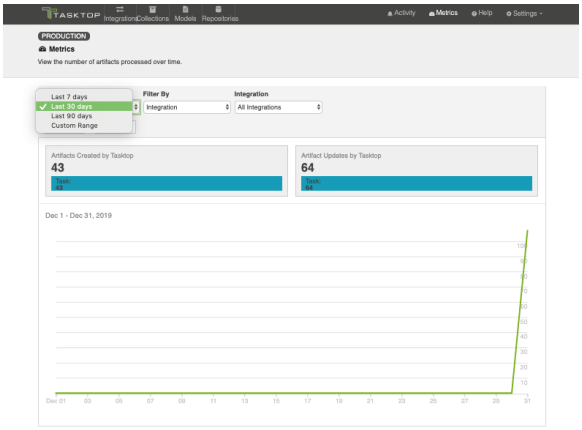
Additionally, users can choose to filter the data above based on

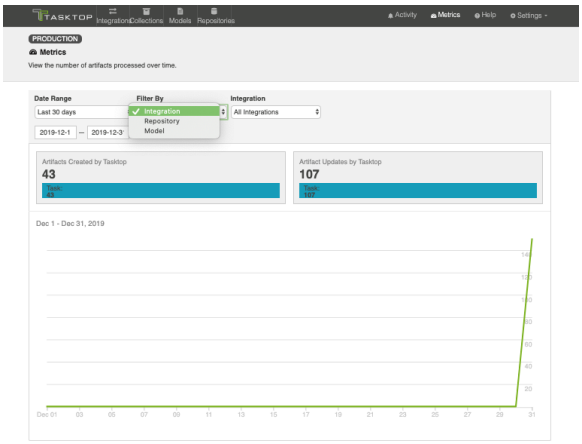
- Date Range
  - Last 7 Days
  - Last 30 Days
  - Last 90 Days
  - Custom Range
- Integration
- Repository
- Model

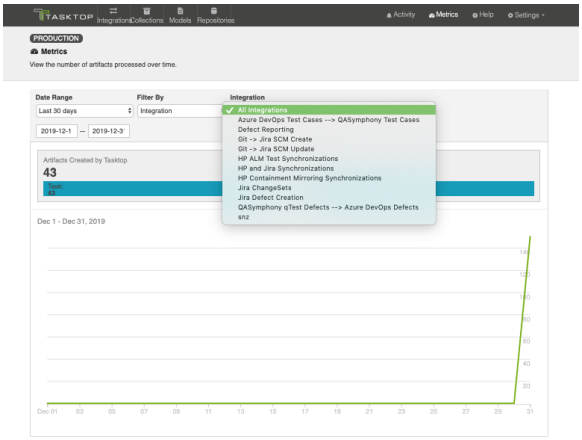
Users can also view tables showing cumulative totals for Artifacts Created, Artifact Updates, and User IDs for each integration and each repository.

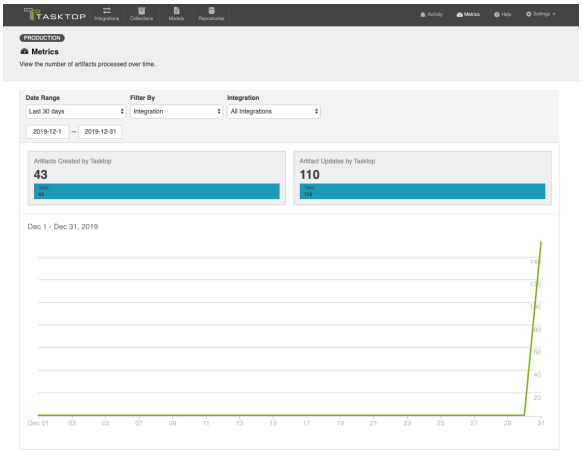
The Artifacts Created and Artifact Updates metrics show cumulative totals since installing Tasktop Integration Hub version 18.2.0.

The User ID metrics shows the number of unique user IDs on artifacts that have flowed through or been updated by Tasktop since installing Tasktop Integration Hub version 18.3.0. This metric can be used to better understand the value and scope of the integration, and is not intended to be used to assess Tasktop usage for licensing purposes (for licensing purposes, please see the Tasktop usage reports).




TASKTOP
Integrations
Collections
Models
Repositories
Activity
Metrics
Help
Settings
PRODUCTION
Metrics
View the number of artifacts processed over time.
Date Range
Last 30 days
2019-12-1
2019-12-3
Filter By
Integration
Repository
Model
Integration
All Integrations
Artifacts Created by Tasktop
43
Artifact Updates by Tasktop
107
Dec 1 - Dec 31, 2019


TASKTOP
Integrations
Collections
Models
Repositories
Activity
Metrics
Help
Settings
PRODUCTION
Metrics
View the number of artifacts processed over time.
Date Range
Last 30 days
Filter By
Integration
Integration
All Integrations
Azure DevOps Test Cases --> QASymphony Test Cases
Defect Reporting
Git -> Jira SCM Create
Git -> Jira SCM Update
HP ALM Test Synchronizations
HP and Jira Synchronizations
HP Containment Mirroring Synchronizations
Jira ChangeSets
Jira Defect Creation
QASymphony qTest Defects --> Azure DevOps Defects
snz
2019-12-1
2019-12-3
Artifacts Created by Tasktop
43
Dec 1 - Dec 31, 2019


TASKTOP
PRODUCTION
Metrics
View the number of artifacts processed over time.
Date Range
Last 30 days
Filter By
Integration
Integration
All Integrations
2019-12-1
2019-12-31
Artifacts Created by Tasktop
43
Artifact Updates by Tasktop
110
Dec 1 - Dec 31, 2019

**Cumulative Artifact Processing Counts**

**Integrations**

| Integration Name | Template | # Artifacts Created | # Artifact Updates | # User IDs ⊕ |
|---|---|---|---|---|
| Azure DevOps Test Cases --> QASymphony Test Cases | Work Item Synchronization | 0 | 0 | 0 |
| Defect Reporting | Enterprise Data Stream | 0 | 0 | 0 |
| Git -> Jira SCM Create | Code Traceability: Create and Relate a Changeset | 0 | 0 | 0 |
| Git -> Jira SCM Update | Code Traceability: Update Existing Work Item | 0 | 0 | 0 |
| HP ALM Test Synchronizations | Work Item Synchronization | 0 | 0 | 0 |
| HP and Jira Synchronizations | Work Item Synchronization | 0 | 0 | 0 |
| HP Containment Mirroring Synchronizations | Container + Work Item Synchronization | 0 | 0 | 0 |
| Jira ChangeSets | Modify via Gateway | 0 | 0 | 0 |
| Jira Defect Creation | Create via Gateway | 0 | 0 | 0 |
| QASymphony qTest Defects --> Azure DevOps Defects | Work Item Synchronization | 0 | 0 | 0 |
| snz | Work Item Synchronization | 43 | 110 | 14 |

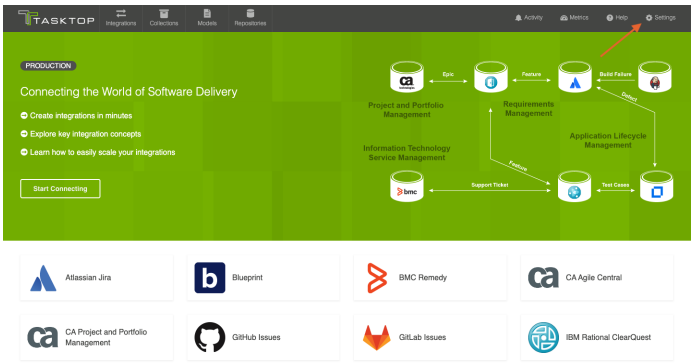
**Repositories**

| Repository Name | # Artifacts Created | # Artifact Updates | # User IDs ⊕ |
|---|---|---|---|
| Azure DevOps | 0 | 0 | 0 |
| Gitea | 0 | 0 | 0 |
| HP ALM | 0 | 0 | 0 |
| JIRA | 0 | 0 | 0 |
| MySQL | 0 | 0 | 0 |
| QASymphony qTest | 0 | 0 | 0 |
| SNOW | 43 | 80 | 1 |
| Zendesk | 0 | 30 | 13 |

# Settings

## Introduction

To access the **Settings** screen, click **Settings** in the upper right corner of your screen.




## General

Under General (Settings), you can access:

- Configuration
- Master Password Configuration
- Storage Settings

## Configuration

The Configuration section allows the administrator to set the change detection intervals of the connected repositories and to create a label for their Tasktop instance identifying the environment name and type (testing or production).

Learn more here.

## Master Password Configuration

*This feature is not applicable to Tasktop Cloud.*

The Master Password is used to encrypt the credentials used in your repository connections and proxy settings, along with any other configuration values that are considered secret, which could include API keys and tokens.

Learn more here.

## Storage Settings

*This feature is not applicable to Tasktop Cloud.*

Tasktop automatically stores operational data to a pre-configured Derby database. This is suitable for evaluation purposes only, and is not supported for production environments. Configuring Tasktop to utilize an external database will enable you to perform frequent back-ups without having to stop Tasktop Integration Hub, and ensure that your Tasktop Integration Hub practices are consistent with your existing disaster and recovery process.

Learn more here.

# Notifications

Under Notifications, you can access:

- Email Notifications

## Email Notifications

To facilitate troubleshooting, you can configure automated email notifications for new errors and issues encountered in Tasktop.

Learn more here.

# License

Under License, you can access:

- License

## License

*This feature is not applicable to Tasktop Cloud.*

A license is required to run the application. Upon initial log-in, you will see that your product is currently un-licensed. You can apply a license and see license details here.

Learn more here.

# Troubleshooting

Under Troubleshooting (Settings), you can access:

- Logging

## Logging

For troubleshooting purposes, Tasktop logs various events that the application performs. You can change the logging level from the Troubleshooting tab.

Learn more here.

# Extensions

Under Extensions (Settings), you can access:

- Extensions
- Key-Value Stores

## Extensions

Extensions add to Tasktop's basic functionality by facilitating processes such as custom data transformations, payload transformations, advanced person reconciliation, and state transitions.

Learn more here.

### Custom Data Transformation

Custom Data Transformation Extensions enable you to map fields to one another which do not have out-of-the-box transforms, and to create custom transforms for comments. You can apply this extension when updating your transform on the Field Configuration screen.

### Payload Transformation

Payload Transformation Extensions enable you to take the payload sent in by your Gateway Collection and transform it into a format that Tasktop can accept. Once you have saved your extension, you can select it on the Gateway Collection screen.

### Person Reconciliation

Tasktop comes with a default person reconciliation strategy ("Copy with Default Matching"), which matches based on name, ID, and/or e-mail. This strategy should cover most use cases. If needed, you can also configure a custom Person Reconciliation Extension to match 'person' fields from one repository to another. You can select the extension on the Person Reconciliation screen during the Collection configuration process.

## State Transition

State Transition Extensions enable you to transition artifacts from one state to another according to a set workflow. The extension can be applied from the State Transition Sash on the Collection Configuration screen.

## Key-Value Stores

Key-Value Stores manage and securely store value mappings and confidential data that can be used in Tasktop Integration Hub's Extensions.

Learn more here.

# Advanced

Under Advanced (Settings), you can access:

- Flow External Workflow Changes
- Move Routes Between Integrations
- Import Artifact Pair Information
- Upgrade Backup Files

## Move Routes Between Integrations

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. You can move routes between integrations on the Advanced Configuration screen.

Learn more here.

## Import Artifact Pair Information

Importing artifact pairs allows Tasktop Integration Hub to know about existing artifact pairs that were created by Tasktop Sync. This prevents duplicate artifacts from being created when you switch from using Tasktop Sync to Tasktop Integration Hub to administer your integrations. Please contact Tasktop Support for additional information on how to use this capability.

## Upgrade Backup Files

Upgrading backup files enables you to download and upload artifact data in cases where integrations were resumed individually during an upgrade. The downloaded data corresponds to artifacts that may have

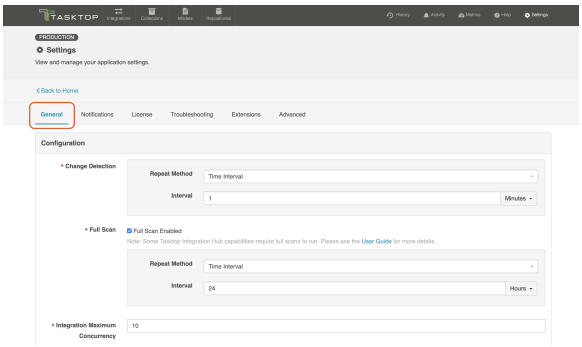
been modified while migrations were still running. These files capture any synchronization activity that occurred on individually running integrations, so that you can ensure no updates are duplicated if restoring from backup.

Learn more here.

# General (Settings)

## Introduction

General (Settings) can be accessed by clicking the **General** tab on the **Settings** screen.




Under **General (Settings)**, you can access:

- Configuration
- Master Password Configuration
- Storage Settings

Under Notifications, you can access:

- Email Notifications

Under License, you can access:

- License

Under Troubleshooting (Settings), you can access:

- Logging

Under Extensions (Settings), you can access:

- Extensions
- Key-Value Stores

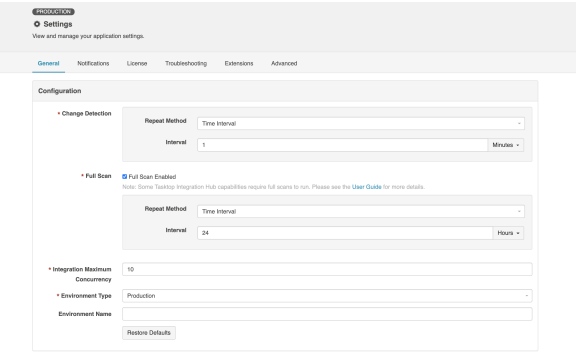
Under Advanced (Settings), you can access:

- Flow External Workflow Changes
- Move Routes Between Integrations

- Import Artifact Pair Information
- Upgrade Backup Files

# Configuration

The **Configuration** section allows the administrator to set the change detection intervals of the connected repositories and to create a label for their Tasktop instance identifying the environment name and type (i.e., testing or production).

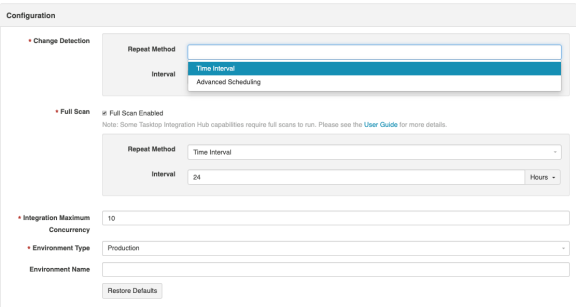



## Change Detection

### Repeat Method

This is the method in which Tasktop detects changed artifacts in your external repositories.

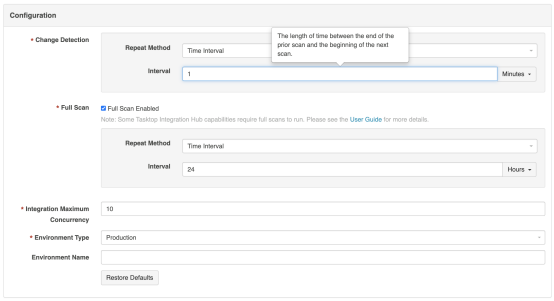
You can configure repeat method to run on a time interval (e.g., 1 minute or 5 minutes) or an advanced schedule using cron expression (e.g., every 30 minutes from 9am-5pm from Monday to Friday).




### Change Detection Interval

This is the time between polling requests made by Tasktop to your external repositories to detect **only changed artifacts**.

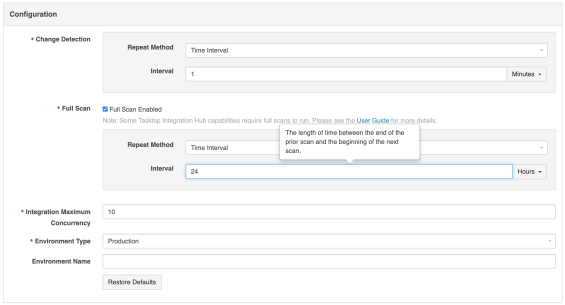
💡 **Note**: This defaults to **1 minute**, but can be customized as desired. This global setting can also be overridden with an integration-specific change detection interval, by updating the Change Detection set tings for that integration.




## Full Scan Interval

This is the time between polling requests to detect changed artifacts, in which all artifacts that have previously synchronized in the integration are scanned.

💡 **Note**: The Full Scan Interval defaults to **24 hours** on the General (Settings) screen, but can be overridden with an integration-specific full scan interval, by updating the Change Detection settings for that integration.




Not all changes to an artifact will register as a change. Some repositories do not mark items as changed — for example, when a relationship is added or an attachment has changed. These changes may **not** be picked up by regular Change Detection, but **will** be picked up by a Full Scan.

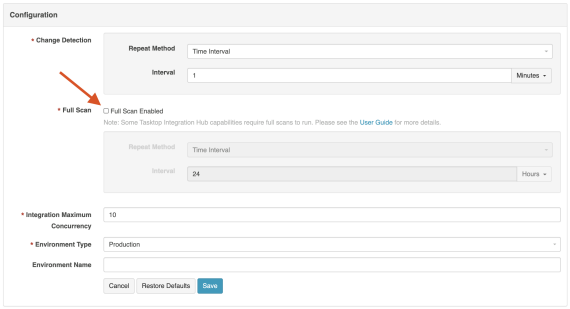
💡 **Tip**: Review our connector docs to determine the type of updates that will require a full scan.

#### Disable 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.

To disable full scan, uncheck the **Full Scan Enabled** box for the desired collection.

💡 **Note**: If you choose to disable full scans, Twinless Artifact Updates will not work and some artifact updates may be missed.




**Process All Artifacts**

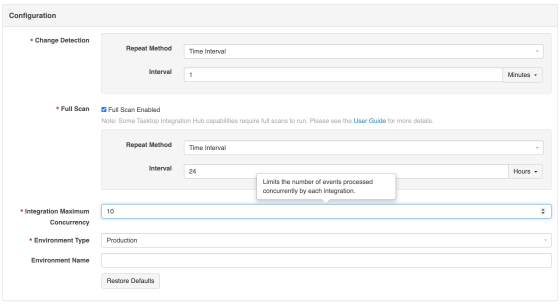
Since the Full Scan only scans artifacts that have previously synchronized, artifacts that are newly eligible for synchronization due to updated artifact filtering or routing will not be picked up by the Full Scan.

These artifacts will only be processed by clicking Process All Artifacts on the Field Flow screen, or when a new integration-eligible change is made to them.

## Integration Maximum Concurrency

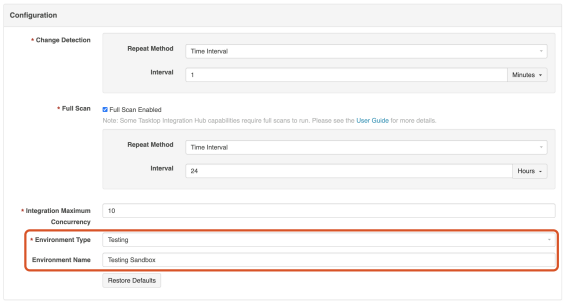
This limits the number of events processed concurrently by each integration. Increasing this value will enable more artifact changes to flow concurrently, whereas decreasing this value will reduce the level of concurrent changes. Changing this value has the potential to affect the load on the end-points of an integration, and may have an adverse effect on performance if set too high.

💡 **Note**: The default setting is **10**, and should be used unless advised otherwise by Tasktop Support.

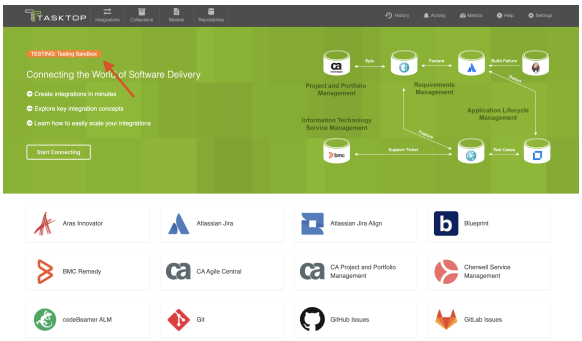



# Environment Type and Name

Tasktop administrators can also set an environment type (testing or production) and name for their instance in the Configuration panel. This will create a label visible in the upper left corner of the screen while navigating throughout the Tasktop UI, to allow users to easily identify which Tasktop instance they are utilizing.



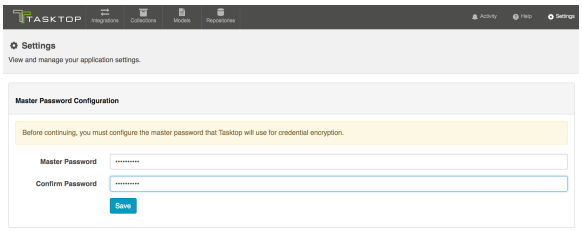

Once set, you will see the environment type and name label displayed in Tasktop.




# Master Password Configuration

***This feature is not applicable to Tasktop Cloud.***

After installation, you will be prompted to set a Master Password.




The Master Password is used to encrypt the credentials used in your repository connections and proxy settings, along with any other configuration values that are considered secret, which could include API keys and tokens.

💡 **Note**: 256 bit AES encryption is used.

Tasktop Integration Hub will automatically use the stored Master Password to decrypt repository credentials.

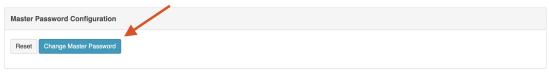
Normally you will not need to re-enter your Master Password. However, if the stored Master Password is missing, or if you'd like to change your Master Password from the General (Settings) screen, you will need to enter your current Master Password.

The Master Password is encrypted and stored separately from the encrypted repository credentials.

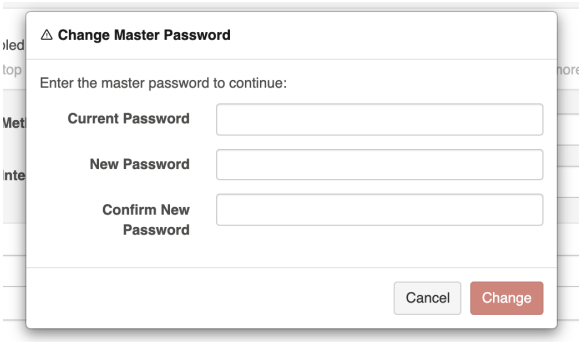
- On **Windows**, the encrypted Master Password is stored in the Windows Registry, encrypted using the Windows Data Protection (DPAPI).
- On **Linux**, the encrypted Master Password is stored in the Home Directory of the User running Tasktop Integration Hub.

If desired, you can change or reset the Master Password from the General (Settings) screen.

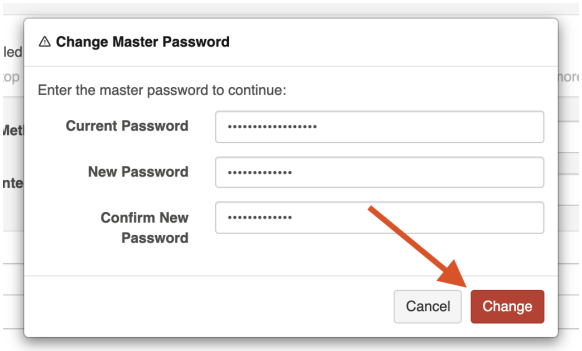
To do this, click **Change Master Password**.




Enter your current Master Password and new Master Password.

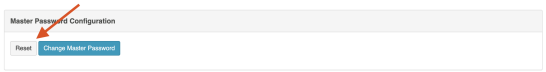



Click **Change** to update the Master Password.




To reset your master password, click **Reset**.

💡 **Note**: If resetting the Master Password, you will not need to enter your current Master Password, but previously encrypted repository passwords will be lost, and must be provided after resetting.



# Storage Settings

*This feature is not applicable to Tasktop Cloud.*

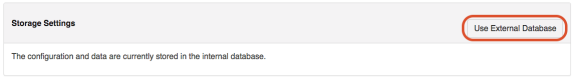
Tasktop automatically stores operational data to a pre-configured Derby database. This is suitable for evaluation purposes only, and is **not** supported for production environments. Configuring Tasktop to utilize an external database will enable you to perform frequent back-ups without having to stop Tasktop Integration Hub, and ensure that your Tasktop Integration Hub practices are consistent with your existing disaster and recovery process.

💡 **Tip**: See our Hardware Requirements to determine which databases are supported for storing operational data.

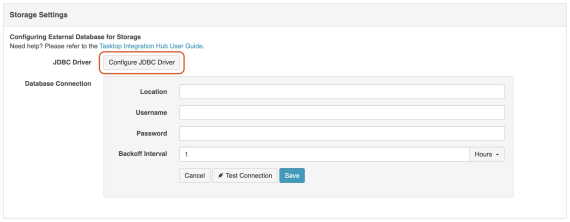
# Migrating Databases

## Internal to External

To migrate your Tasktop operational data from the internal database to an external database, click **Use External Database**.




Next, click **Configure JDBC Driver** to select the JDBC driver for your database.




To download the JDBC driver:

### PostgreSQL Server

The JDBC driver for PostgreSQL can be downloaded from the PostgreSQL download site.

### Microsoft SQL Server

The JDBC driver for Microsoft SQL Server can be downloaded from the Microsoft support site.

💡 **Note**: Tasktop currently supports the 7.0.0.jre8 driver version.

### 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.

### MySQL Server

The JDBC driver for MySQL can be downloaded from the MySQL download site.

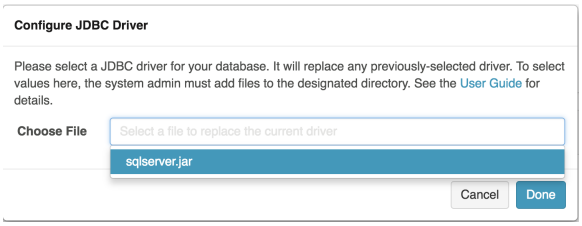
⚠ **Note**: Upon downloading the JDBC driver from the MySQL download site, select **Platform Independent** to download the correct file.

To upload the JDBC driver to Tasktop, a system administrator (a user with file system access to the machine that hosts Tasktop) must extract the ***.jar** file from the downloaded driver file and add the file to the designated directory:

- On **Windows**, the default folder is `C:\ProgramData\Tasktop\jdbc-drivers`
- On **Linux**, the `jdbc-drivers` folder can be found in the Tasktop installation directory

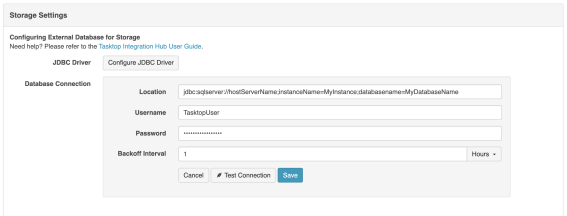
💡 **Note**: If needed, the user can change the location in which Tasktop looks for the files. This is done by changing the system property `jdbc.libraries.path`

Once the JDBC driver is uploaded, select it from the **Choose File** field on the Configure JDBC Driver pop-up.




Next, fill out the Database Connection credentials — enter the location, username, and password.

⚠ **Note**: Authentication credentials **must** be in SQL server authentication mode (i.e., mixed-mode with SQL credentials). Windows authentication mode is **not** supported.




Location formats are as follows:

### PostgreSQL

*jdbc:postgresql://hostServerName:postgreSqlServerPort/MyDatabaseName*

💡 **Note**: If you use a custom schema, you will need to add **?currentSchema=tasktop** to the URL (e.g., jdbc:postgresql://example.com:5432/dbName?currentSchema=tasktop).

### Microsoft SQL Server

*jdbc:sqlserver://hostServerName;instanceName=MyInstance;databasename=MyData*

### Oracle

*jdbc:oracle:thin:@hostServerName:oracleServerPort/SID*

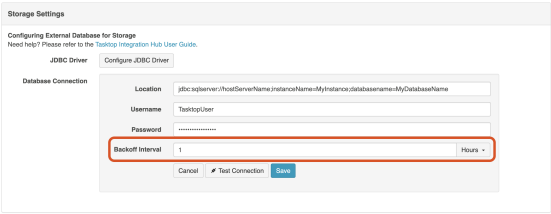
### MySQL Server

*jdbc:mysql://hostServerName:mysqlServerPort/MyDatabaseName*

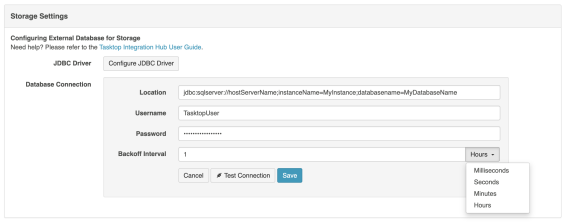
If you'd like, you can also update the **Backoff Interval** setting.

The **Backoff Interval** is the time Tasktop will wait after a database connection failure (e.g., invalid username or password) before retrying the connection. This feature is especially useful for databases with a lockout or brute force policy configured.

⚠️ **Note**: While backoff is in effect, processing artifacts will display an error and some operations may not work (e.g., you may be automatically redirected to the Settings screen). Once the backoff interval expires, artifacts will resume processing and operations will return to normal.

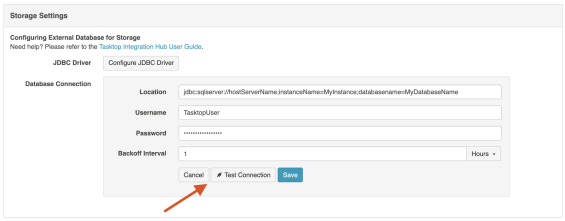



The backoff interval defaults to **one hour**, but can be customized as desired.

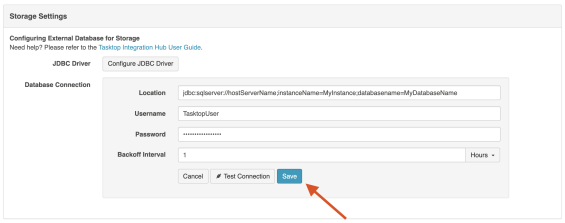



After you've added your database connection credentials, click **Test Connection** to confirm that your credentials have been accepted by Tasktop.

💡 **Note**: If backoff is in effect, the Test Connection button will continue to work so you can test and save updated credentials.

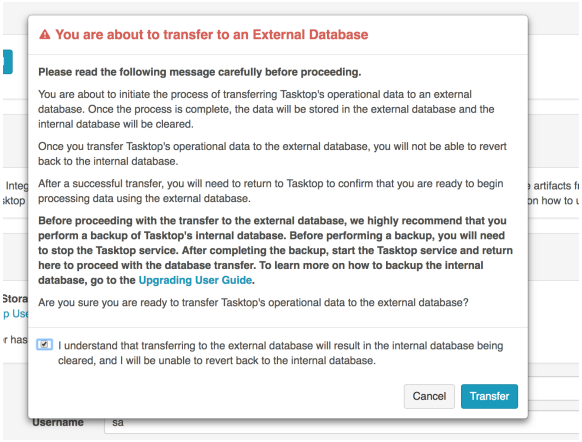



Once confirmed, click **Save**.

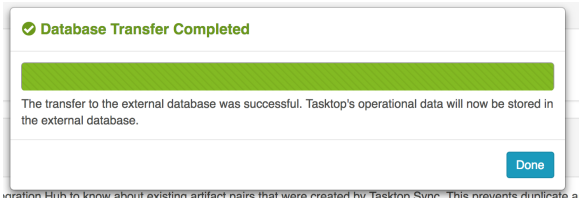



A warning message will appear telling you that you are about to transfer to an External Database. Review the entire message, **ensuring that you have performed the recommended data back-up.**

If you'd like to continue the transfer, click the checkbox and then click **Transfer**.




A **Database Transfer Completed** message will appear once the transfer is complete, informing you that your operational data has been successfully transferred from Tasktop's internal database to your own external database.




# External to External

> ⚠ If you'd like to migrate your data from one external database to a different external database, **yo u will need to manually transfer the data from the current database to the new target database.** I

f you do not manually transfer the data, Tasktop will not work properly once you switch to the target database settings. **Tasktop will not automatically transfer this data for you.**

If you are simply updating the location or credentials of your current external database and will continue using the same database, you do not need to transfer any data. Tasktop will continue to work properly.

Moving between databases of the same type...

If you are migrating to a database of the same type (e.g., moving from one MySQL database to a different MySQL database), follow the instructions below:

1. Transfer data from the old database to the new database.
2. Update the Location, Username, and/or Password fields in the Database Connection section.
3. Click **Test Connection** and then **Save**.
4. A warning message will then appear, ensuring you have taken all necessary steps. After reviewing the message, click the checkbox and then **Save**.

Moving between databases of different types...

If you are migrating to a database of a different type (for example, moving from a MySQL database to an Oracle database), follow the instructions below:

1. Create a new empty database in the new database.
2. Stop Tasktop.
3. Manually replace the jdbc driver jar in `<program data>/Tasktop/drivers` with the correct driver for the new database (not in `<program data>/Tasktop/jdbc-drivers`, because the new driver cannot be selected in the UI), and make sure it is named `database-driver.jar`.
4. Manually edit `<program data>/Tasktop/db/tasktop-db.json` with the URL and credentials for the new database.
5. Start Tasktop.
6. Tasktop will create new empty tables in the new database.
7. Stop Tasktop.
8. Copy all the data from the tables in the old database to the tables in the new database, except the tables `DATABASECHANGELOG` and `DATABASECHANGELOGLOCK` (copying data for these two tables will cause errors).
9. Start Tasktop.

# If your Database Transfer Fails or is Aborted

If your database transfer fails or is aborted, Tasktop will continue to use its internal database to store operational data. The internal database is not cleared until a successful transfer is completed, so you should not notice any change in performance.

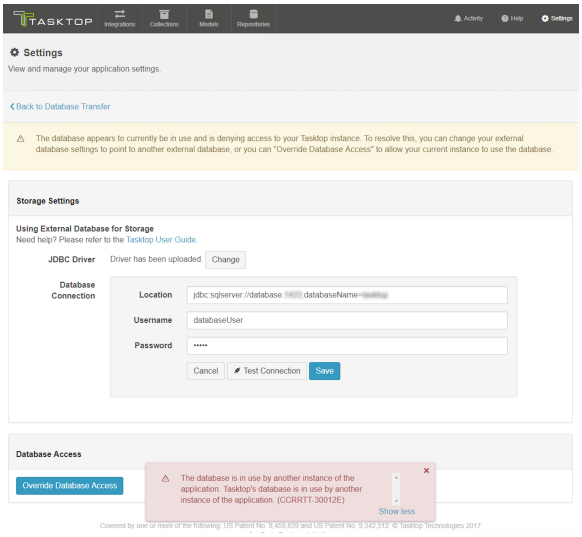
However, we do recommend reviewing the external database and clearing any data and tables that were created as part of the failed data transfer before starting the transfer process again.

# Overriding Database Access

In order to prevent risk of collisions, duplicates, and other errors, Tasktop has functionality to ensure that multiple Tasktop instances cannot run on the same operational database.

If you connect your instance to a database that is already in use by Tasktop (this is **not** recommended), upon start-up of the new instance, the prior instance will lose database access and stop processing events. When you login to the prior instance, you will see an error message prompting you to either update your credentials to connect to a different database, or to override database access. If you override database access, this means that the other instance of Tasktop will lose access to that database.

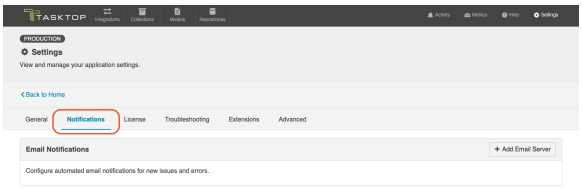
When overriding, be sure to confirm that no other Tasktop instance is using the database before moving forward. If another Tasktop instance is actively using the database, it is recommended that you shut down the other instance of Tasktop before proceeding.



# Notifications

## Introduction

Notifications can be accessed by clicking the **Notifications** tab on the **Settings** screen.




Under **Notifications**, you can access:

- Email Notifications

Under General (Settings), you can access:

- Configuration
- Master Password Configuration
- Storage Settings

Under License, you can access:

- License

Under Troubleshooting, you can access:

- Logging

Under Extensions, you can access:

- Extensions
- Key-Value Stores

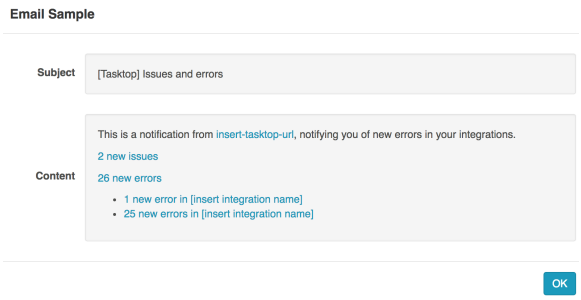
Under Advanced, you can access:

- Flow External Workflow Changes
- Move Routes Between Integrations
- Import Artifact Pair Information
- Upgrade Backup Files

# Email Notifications

To facilitate troubleshooting, you can configure automated email notifications for new errors and issues encountered in Tasktop.

Emails will contain a count of new issues and errors (excluding ignored errors) since the last email notification, and a link to the Activity screen to view the errors/issues. Emails will be sent only if a new error or issue occurs.

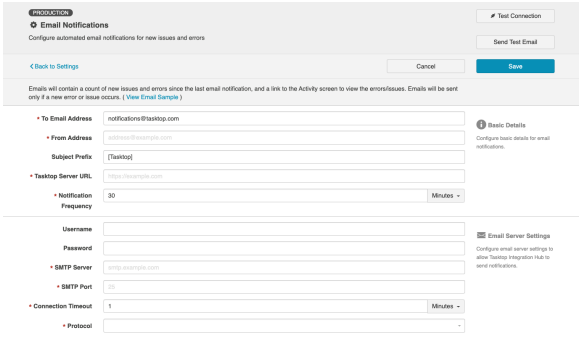



## Configuring Email Notifications

## Tasktop Hub On-prem

To configure email notifications click **+Add Email Server**.


This will bring you to the **Email Notifications** screen.




The form requires that the following fields be filled out:

### Basic Details

- **To Email Address**: The email addresses that will receive the notifications. You can add multiple email addresses to this field.
- **From Email Address**: The email address listed in the 'sender address' (or 'from') field of notification emails sent by Tasktop. In many cases, this will match the email whose settings are configured in the 'Email Server Settings' section below, though a different email (such as no-reply@email.com) can be configured here. If a user were to hit 'reply' on an email notification, this is the email the reply would be sent to.
- **Subject Prefix (optional)**: The prefix appended to the subject line of the Error Notification emails. This defaults to [Tasktop] but can be edited if needed. This can help users filter email notifications from Tasktop based on prefix.
- **Tasktop Server URL**: The URL used to access your instance of Tasktop. This is used to construct links to errors and issues in the notification emails.
- **Notification Frequency**: The frequency of email notifications. Emails will contain a count of new issues and errors since the last notification and will only send if a new error or issue has occurred since the prior email.

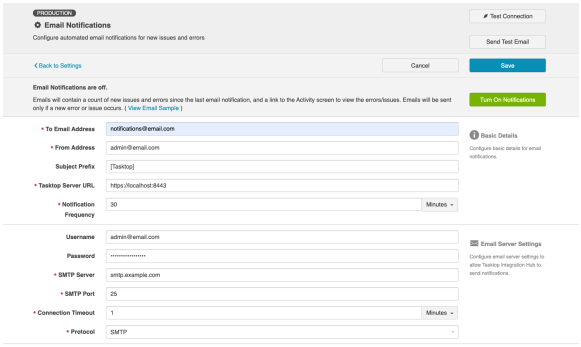
### Email Server Settings

These are the email server settings that allow Tasktop to send notifications.

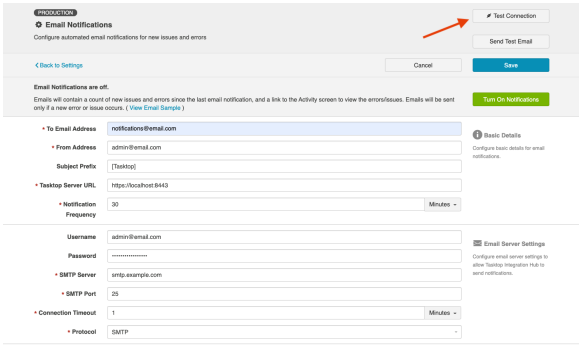
- **Username (optional)**: Username for the authenticated SMTP server.
- **Password (optional)**: Password for the authenticated SMTP server.
- **SMTP Server**: The SMTP host name of your mail server.
- **SMTP Port**: The SMTP Port number to use.
  - If Protocol = SMTP, the value for this will typically be 25.
  - If Protocol = SMTPS, the value for this will typically be 465.
  - If Protocol = SMTP_STARTTLS, the value for this will typically be 587, but can also be port 25.
- **Connection Timeout**: Specifies the maximum period, in seconds, that establishing an email server connection is permitted to take. This defaults to 60 seconds, which should cover most scenarios.
- **Protocol**
  - **SMTP**: Basic unencrypted SMTP Protocol.
  - **SMTPS**: A more advanced, encrypted SMTP Protocol (SMTP Secure), which will perform server certificate validation.
  - **SMTP_STARTTLS**: A modern protocol that wraps the unencrypted SMTP protocol in TLS (formerly known as SSL encryption), and will perform server certificate validation. This will attempt the STARTTLS wrapper, but if it is not supported by the server, the client will fall back to basic SMTP.

💡 **Note**: Google email users should select SMTP_STARTTLS.

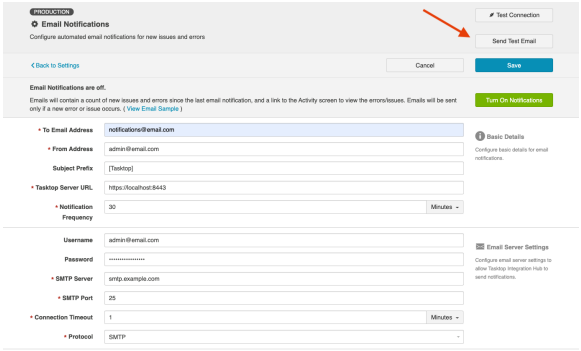
Here's an example of a filled in form:



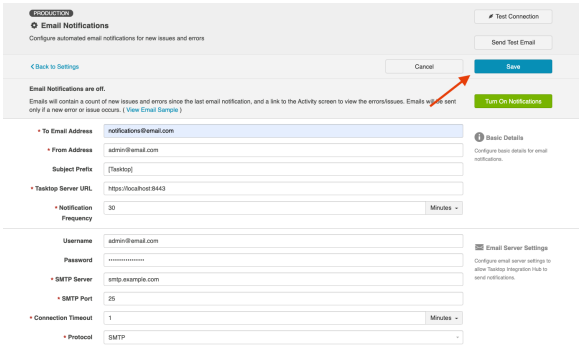

You can test your email server settings by clicking **Test Connection**.



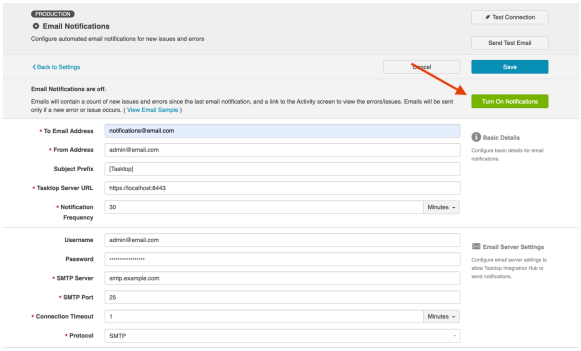

Or, send a test email by clicking **Send Test Email**.



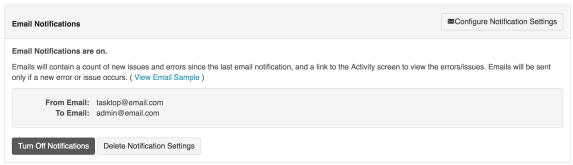

Once settings are filled in and the connection has been tested, click **Save** to save your settings.




Click **Turn On Notifications** to enable email notifications.



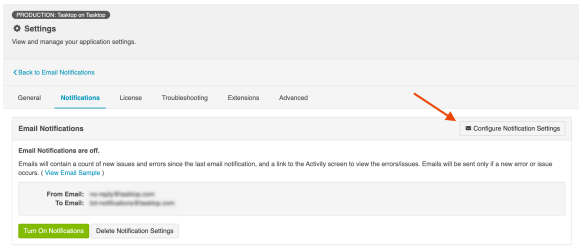

Once saved, you can turn email notifications on or off and delete the notification settings from the Notifications screen. You can also click **Configure Notification Settings** to modify your existing settings:



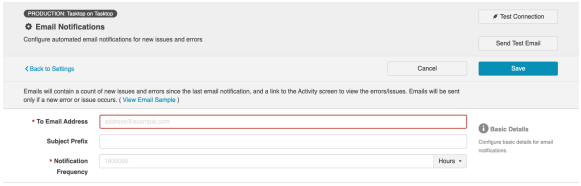

💡 **Note**: If an email notification fails, an issue will be surfaced on the Activity screen in Tasktop.

# Tasktop Hub Cloud

To configure email notifications, click **Configure Notification Settings**.




This will bring you to the **Email Notifications** screen.



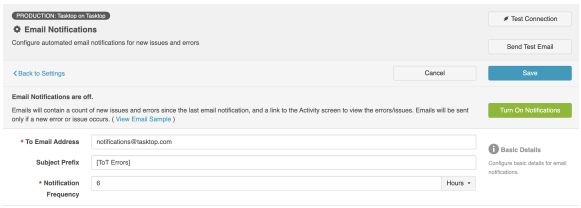

The form requires that the following fields be filled out:

### Basic Details

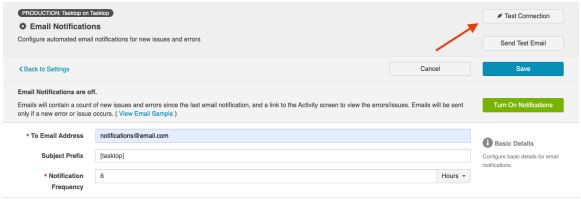
- **To Email Address**: The email address that will receive the notifications. This field is limited to one email address.

- **Subject Prefix (optional)**: The prefix appended to the subject line of the Error Notification emails. This defaults to **[Tasktop]** but can be edited if needed. This can help users filter email notifications from Tasktop based on prefix.
- **Notification Frequency**: The frequency of email notifications. Emails will contain a count of new issues and errors since the last notification and will only send if a new error or issue has occurred since the prior email.

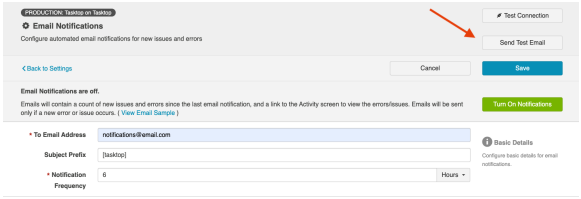
Here's an example of a filled in form:



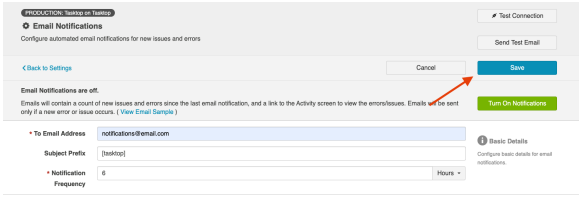

You can test your email server settings by clicking **Test Connection**.



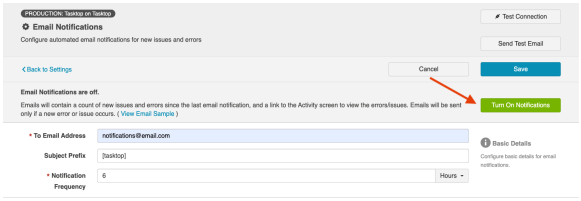

Or, send a test email by clicking the **Send Test Email** button.




Once settings are filled in and the connection has been tested, click **Save** to save your settings.

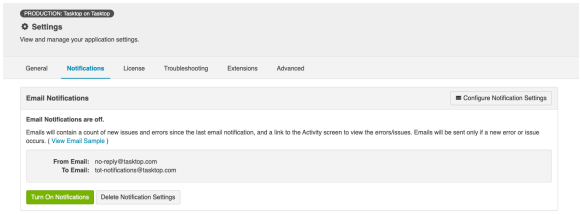



Click **Turn On Notifications** to enable email notifications.




Once saved, you can turn email notifications on or off and delete the notification settings from the Notifications screen.

You can also click **Configure Notification Settings** to modify your existing settings:

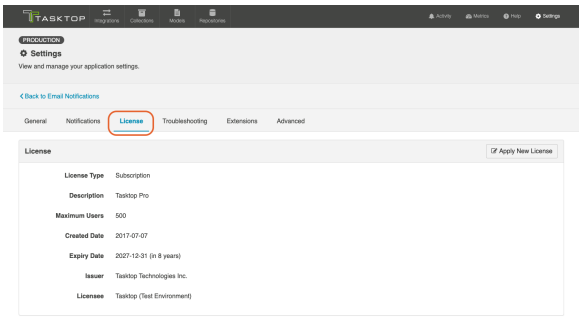



**Note**: If an email notification fails, an issue will be surfaced on the Activity screen in Tasktop.

# License

## Introduction

You can access your License details by clicking the **License** tab on the **Settings** screen.




Under **License**, you can access:

- License

Under General, you can access:

- Configuration
- Master Password Configuration
- Storage Settings

Under Notifications, you can access:

- Email Notifications

Under Troubleshooting, you can access:

- Logging

Under Extensions, you can access:

- Extensions
- Key-Value Stores

Under Advanced, you can access:

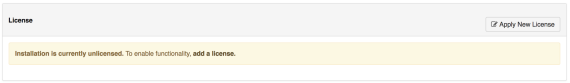
- Flow External Workflow Changes
- Move Routes Between Integrations
- Import Artifact Pair Information

- Upgrade Backup Files

## License

*This feature is not applicable to Tasktop Cloud.*

A license is required to run the application. Upon initial log-in, you will see that your product is currently un-licensed:

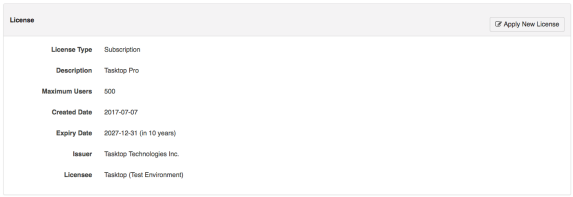



Click **Apply New License** to enter your license.

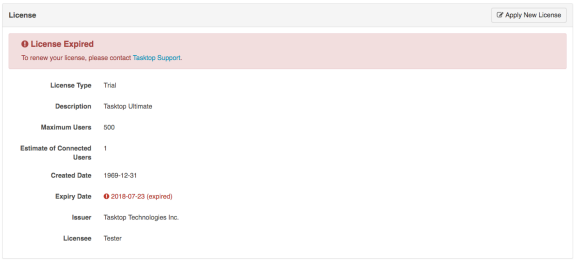
The Master Password must be set and the License must be entered before the application can be used.

On the License panel, you will see the following information:

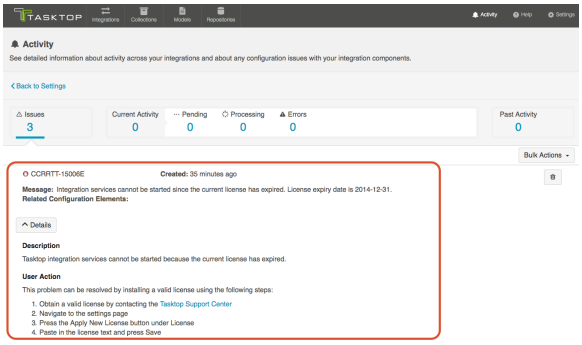
- License Type
- Description
- Maximum Users
- Created Date
- Expiration Date
- Issuer
- Licensee




You will also see a warning if your license is expired:




Should your license expire, in addition to seeing a warning on the License screen, you'll also see that an issue is surfaced on the Activity screen:



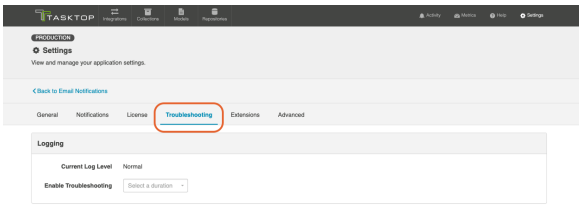

When your license is expired, you can still navigate within the Tasktop UI, but your integrations will be stopped from running. Note that though they will still display the **Run** or **Stopped** state they were in at the time your license expired, no artifacts will process in an integration until a new license is applied.

💡 **Note**: Please consult your license agreement or contact your account representative if you have any questions about your license settings or usage policy.

# Troubleshooting (Settings)

## Introduction

Troubleshooting (Settings) can be accessed by clicking the **Troubleshooting** tab on the **Settings** screen.




Under **Troubleshooting**, you can access:

- Logging

Under General (Settings), you can access:

- Configuration
- Master Password Configuration
- Storage Settings

Under Notifications, you can access:

- Email Notifications

Under License, you can access:

- License

Under Extensions, you can access:

- Extensions
- Key-Value Stores

Under Advanced, you can access:

- Flow External Workflow Changes
- Move Routes Between Integrations
- Import Artifact Pair Information

- Upgrade Backup Files

# Logging

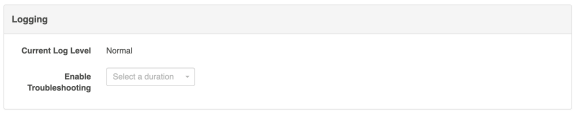
For troubleshooting purposes, Tasktop logs various events that the application performs.

There are two logging levels available:

## Normal Logging

This type of logging is sufficient for most scenarios.

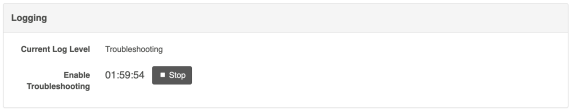
Updating the logging levels immediately changes the logging granularity. Tasktop does not need to be restarted for the change to take effect.




## Troubleshooting Logging

This setting provides more detailed logs. Due to the large volume of logs created during Troubleshooting logging, this option has a time limit with a maximum of 24 hours. If Troubleshooting level is selected, the Normal logging level can be enabled at any time by clicking **Stop Troubleshooting Now**.

Updating the logging levels immediately changes the logging granularity. Tasktop does not need to be restarted for the change to take effect.



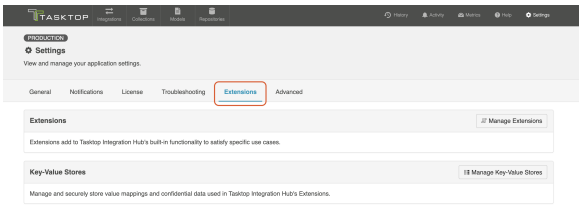

## Downloading Logs

Please reference the Troubleshooting page for instructions on downloading the logs as part of the Support and Usage Report.

# Extensions (Settings)

## Introduction

Extensions can be accessed by clicking the **Extensions** tab on the **Settings** screen.




Under **Extensions**, you can access:

- Extensions
- Key-Value Stores

Under General (Settings), you can access:

- Configuration
- Master Password Configuration
- Storage Settings

Under Notifications, you can access:

- Email Notifications

Under License, you can access:

- License

Under Troubleshooting (Settings), you can access:

- Logging

Under Advanced (Settings), you can access:

- Flow External Workflow Changes
- Move Routes Between Integrations
- Import Artifact Pair Information
- Upgrade Backup Files

# Extensions

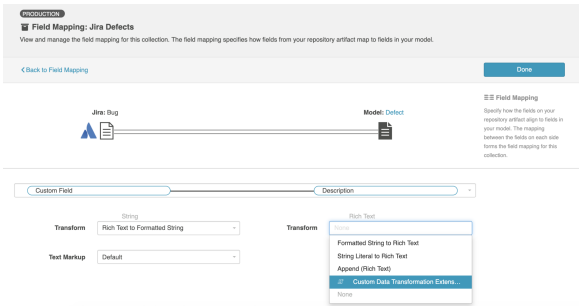
**Extensions** add to Tasktop's basic functionality by facilitating processes such as custom data transformations, payload transformations, advanced person reconciliation, and state transitions.

💡 **Note**: Extensions are written in JavaScript, or more specifically ECMAScript.

### Custom Data Transformation

Custom Data Transformation Extensions enable you to map fields to one another which do not have out-of-the-box transforms, and to create custom transforms for comments.

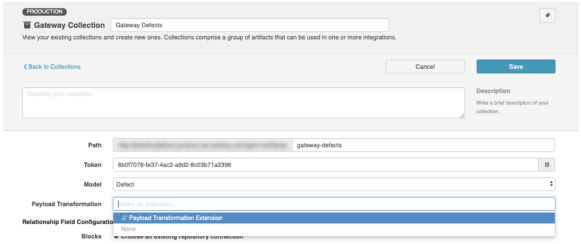
You can apply this extension when updating your transform on the Field Configuration screen.




### Payload Transformation

Payload Transformation Extensions enable you to take the payload sent in by your Gateway Collection and transform it into a format that Tasktop can accept.

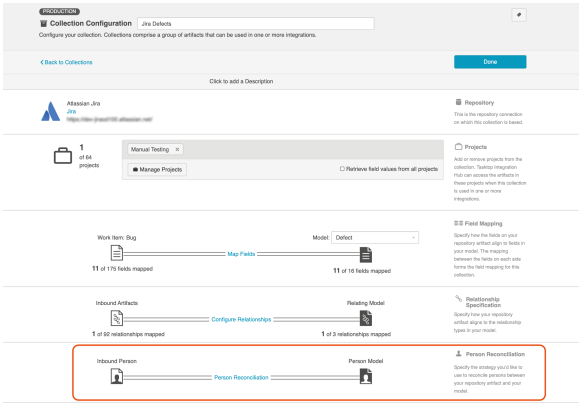
Once you have saved your extension, you can select it on the Gateway Collection screen.




### Person Reconciliation

Tasktop comes with a default person reconciliation strategy (**Copy with Default Matching**), which matches based on name, ID, and/or email. This strategy should cover most use cases. If needed, you can also configure a custom Person Reconciliation Extension to match **person** fields from one repository to another.

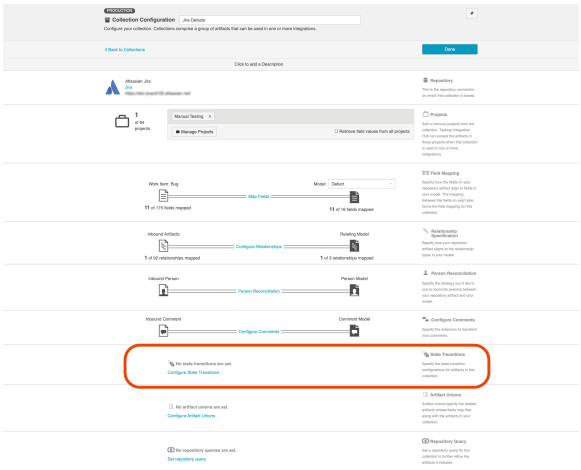
You can select the extension from the Person Reconciliation sash on the **Collection Configuration** screen.



### State Transition

State Transition Extensions enable you to transition artifacts from one state to another according to a set workflow.

The extension can be applied from the State Transition sash on the **Collection Configuration** screen.

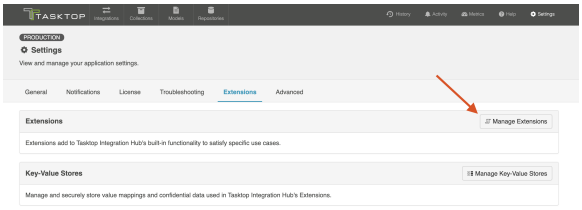


## Creating a New Extension

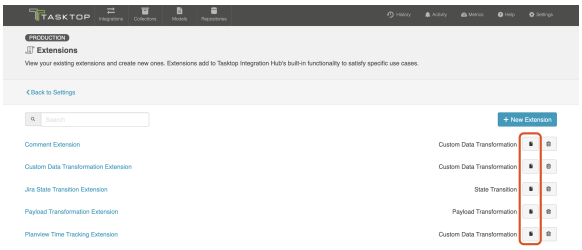
You can create and save custom extensions for use in your integrations on the **Extensions** screen. Extensions are created with a name and optional description so that they can be centrally managed and reused if needed.

💡 **Note**: Fields that are not mapped to the model are not retrieved by Tasktop, and therefore are not available to be used in an extension. If fields are needed for scripting purposes, please map those fields to the model.

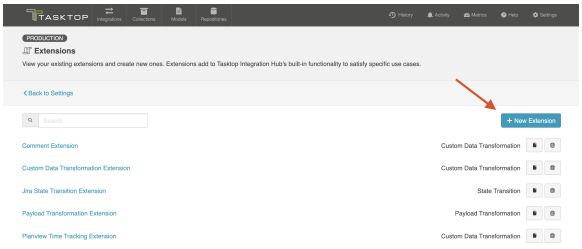
To create and edit your extensions, click **Manage Extensions**.



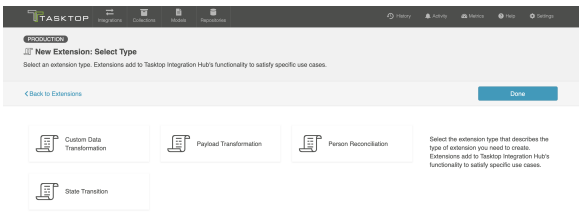

You can copy an existing extension by clicking **Copy** from the Extensions list.




Click **New Extension** to create and customize an extension.

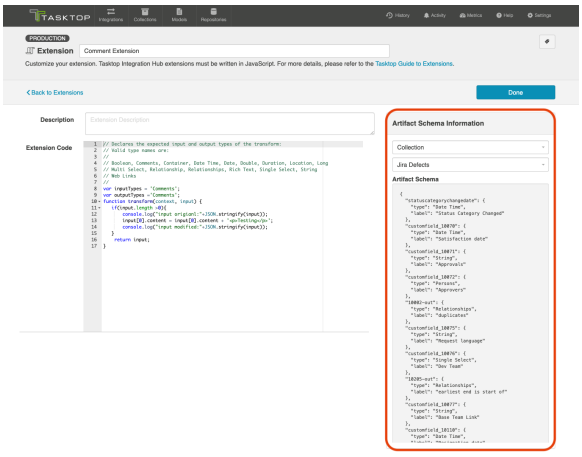



Then, select one of the following extension types: **Custom Data Transformation**, **Payload Transformation**, **Person Reconciliation**, **State Transition**.



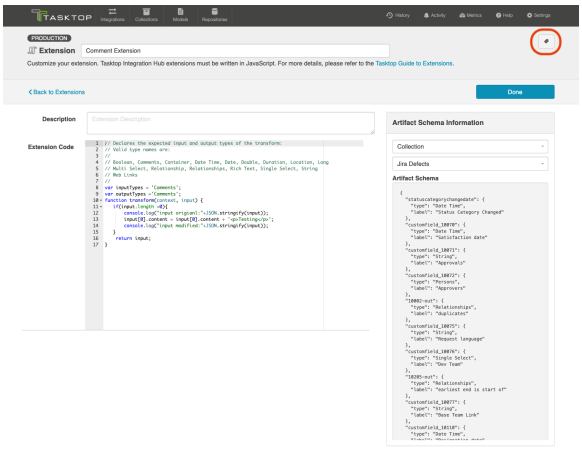

After choosing your extension type and selecting the collection or model, you can view the schemas of the artifact, comment, or person in the Artifact Schema Information section.

This section provides you with useful schema information when composing extensions.



## Viewing Associated Configuration Elements

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



# Key-Value Stores

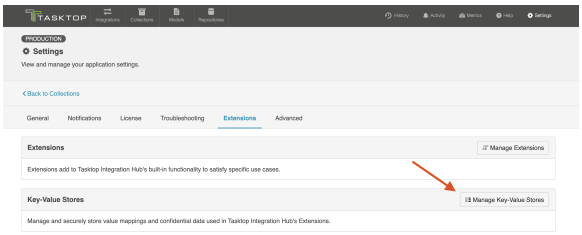
*See the Tasktop Editions table to determine if your edition contains Key-Value Store functionality.*

**Key-Value Stores** enable you to securely store and manage sensitive data and value mappings. Using key-value stores instead of inlining the data in the extensions reduces the size, complexity, and maintenance of extensions.

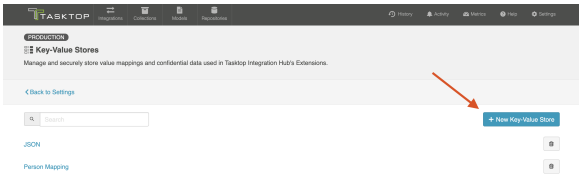
## Creating a New Key-Value Store

💡 **Note**: Access via the provided JavaScript API is read-only. Stores can only be created, updated, and deleted using the user-interface and import functionality.

To create a new key-value store, navigate to the **Extensions** tab and click **Manage Key-Value Stores**.

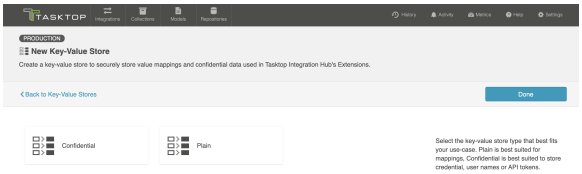



Click **+ New Key-Value Store.**



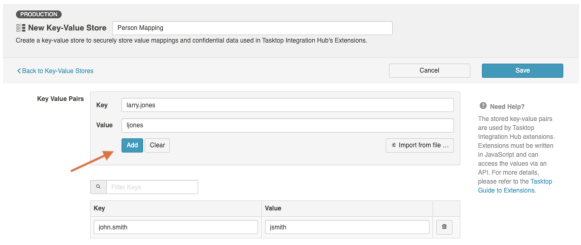

Select the Key-Value store type that best suits your use-case.

- **Confidential:** Enables you to encrypt your key-value pairs.
- **Plain:** Enables you to store your key-value pairs in plain text.

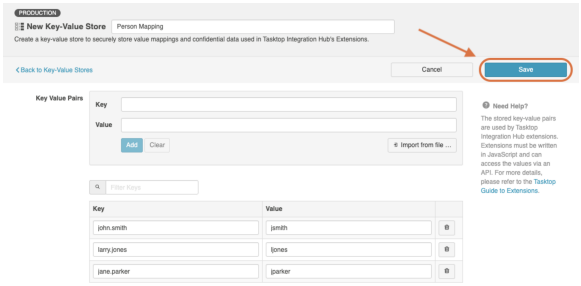



After you have selected the Key-Value store type, you can add each key-value pair individually, or you can import key-value pairs using a .**csv** or .**json** file.

💡 **Note**: Keys in a key-value store are case-sensitive and must match exactly.



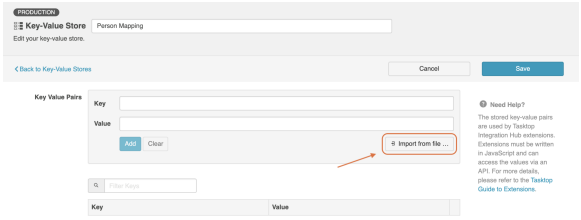

Once you have finished adding your key-value pairs, click **Save** and **Done** to save your changes.




# Importing Files to a Key-Value Store

Tasktop Integration Hub allows you to import key-value pairs to your key-value store using .**csv** or .**json** files.

To import your key-value pairs, click **Import from file** and select the .**json** or .**csv** file you'd like to import.




To ensure that your .json or .csv files are imported successfully, please use the following format:

### .json

```
{
"jsmith@email.com":"John Smith",
"ljones@email.com":"Larry Jones",
"mbrown@email.com":"Mary Brown"
}
```

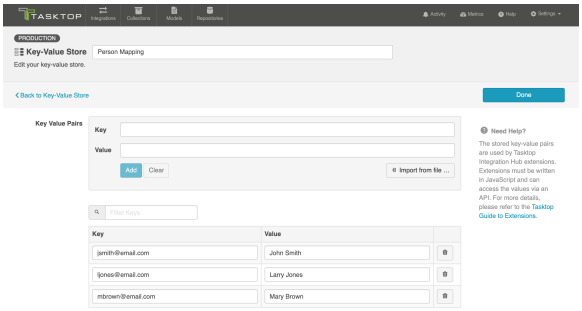
⚠ **Note**: If your .json file contains a duplicate key, an error will appear upon importing.

### .csv

```
"jsmith@email.com", "John Smith"
"ljones@email.com", "Larry Jones"
"mbrown@email.com", "Mary Brown"
```

⚠ **Note**: If your .csv file contains a duplicate key, an error will appear upon importing.

After you have selected the file you'd like to import and it has been imported successfully, the key-value pairs will be displayed in your key-value store.




# Accessing Individual Values of a Key-Value Store

Tasktop Integration Hub provides a JavaScript API to access the values of a store. All types of key-value stores allow direct access to values.

The API is used as follows:

```
var tokenValue = store.retrieveValue('EngOps Credentials', 'Build Server API token');
```

# Accessing All Pairs of a Key-Value Store

The provided JavaScript API allows the retrieval of all pairs of a store. This is especially useful when using the stored pairs as a lookup table.

The API is used as follows:

```
var pairs = store.retrievePairs('Project to Product mapping');
```

Each store is limited to 3000 key-value pairs. For lookup and other purposes, the pairs of multiple stores can be joined in the extension as follows:

```
var pairs = store.retrievePairs('Project to Product mapping');
var morePairs = store.retrievePairs('Value-Stream mapping');

var merged = Object.assign(pairs, morePairs);
```

💡 **Note**: Confidential key-value stores do not allow access to all pairs and can only be accessed by providing the key for individual values.

# Key-Value Store API Reference

- `store` - The globally-visible object providing the key-value store API.
- `store.retrievePairs(store)` - Provides an object with all the keys as properties and their values as strings. If the store cannot be found, a 'NotFoundException' is thrown.
- `store.retrieveValue(store, key)` - Provides the string value associated to the specified key. If the store or key cannot be found, a 'NotFoundException' is thrown.

# Technical Guide to Extensions

Extensions add to Tasktop's built-in functionality to satisfy specific use cases, such as:

- Performing state transitions incorporating business logic
- Enabling custom data transformations between fields
- Defining person reconciliation strategies between repositories
- Transforming payloads sent to Gateway collections into a format Tasktop can accept

In the following sections, you will find technical implementation details about each extensions type, example extensions, troubleshooting extensions, and how to access web resources and object properties.

# State Transitions

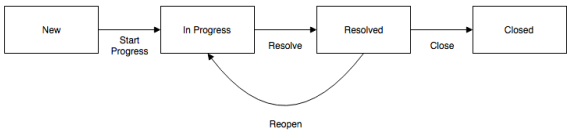
**State Transitions** are used to transition an artifact from one status to another. To illustrate, we use the fictitious example of an artifact of type Defect with the following status values:

- New
- In Progress
- Resolved
- Closed

The status of a Defect cannot be modified directly. In this example, to move a defect from status **New** t o **In Progress**, the **Start Progress** transition is used.

Sometimes multiple status transitions are required. For example, to move a defect from **New** to **Closed**, the following transitions are used in sequence **Start Progress**, **Resolve**, **Close**.

The following diagram shows how state transitions are used to move a defect from one status to another:




## Configuring State Transitions with Extensions

To perform state transitions, an extension can be used. Add a state transition extension from the Extensions screen, accessible from Settings. Once added, the extension can be applied from the State Transition sash on the Collection Configuration screen.

💡 **Note**: Tasktop also provides functionality to configure state transitions using a transition graph. The transition graph is the recommended strategy, as it allows you to configure the state transitions directly within Tasktop's UI.

## Authoring State Transition Extensions

State transition extensions are defined by a single function:

```
function transitionArtifact(context,transitions)
```

This function can return a single transition.

For a given artifact, the extension may be called multiple times. Each time the extension is called, the transition that it returns is performed. State transition extensions are called repeatedly until they return undefined, indicating that no more transitions are needed.

To prevent errors, extensions are not called again if they cause an artifact to transition to the same status more than once.

A simple state transition extension could look something like this:

```
function transitionArtifact(context,transitions) {

    if (context.sourceArtifact.status === 'Resolved' && context.targetRepositoryArtifact.status !== 'Resolved')
{
        var transition = findTransitionWithLabel(transitions,'Resolve');
        transition.attributes.resolution = 'Fixed';

        return transition;

    }

}

function findTransitionWithLabel(transitions, label) {

    for each(var transition in transitions) {

        if (transition.label === label) {

            return transition;

        }

    }

}
```

Two parameters are passed to the `transitionArtifact` function:

- `context` - A context object that provides state that the extension can use to determine which transitions are needed
  - `context.sourceArtifact` - A JavaScript object representation of the source artifact, whose structure matches the model configured in the integration
  - `context.targetRepositoryArtifact` - A JavaScript object representation of the target artifact, whose structure matches the structure of the artifact in the repository
- `transitions` - An array of transition objects

Below is an example of a `context` with a target artifact from Jira:

```
{
    "sourceArtifact": {
        "summary": "a summary value",
```

```
        "priority": "Critical",
        "status": "Done"
    },
    "targetRepositoryArtifact": {
        "issuetype": "Bug",
        "components": null,
        "timespent": null,
        "formattedid": "TPC-144",
        "timeoriginalestimate": null,
        "project": "Test Project C",
        "description": null,
        "fixVersions": null,
        "resolution": null,
        "customfield_11500": null,
        "api-id": "JIRA",
        "attachment": null,
        "resolutiondate": null,
        "id": 14400,
        "summary": "a summary value",
        "watches": null,
        "created": "2016-09-23T15:22:20.000+0000",
        "$closed": false,
        "reporter": "****",
        "priority": "Critical",
        "labels": null,
        "revision": null,
        "customfield_11601": null,
        "customfield_11600": null,
        "customfield_11501": null,
        "environment": null,
        "customfield_11504": null,
        "customfield_11602": null,
        "timeestimate": null,
        "versions": null,
        "duedate": null,
        "web-links": null,
        "location": "http://jira.example.com/browse/TPC-144",
        "assignee": null,
        "worklog": null,
        "updated": "2016-09-23T15:22:20.000+0000",
        "status": "To Do"
    }
}
```

Each transition object in the array appears as follows:

```
 {
  id: 'an-id',
  label: 'A Label'
  attributes: {
    first-attribute: null,
    ...
  }
}
```

For example, transitions corresponding to the Jira artifact example above are as follows:

```
[{
    "attributes": {
        "project": "Test Project C",
        "issuetype": "Bug"
    },
    "id": "11",
    "label": "To Do"
}, {
    "attributes": {
        "project": "Test Project C",
        "issuetype": "Bug"
    },
    "id": "21",
    "label": "In Progress"
```

```
}, {
    "attributes": {
        "project": "Test Project C",
        "issuetype": "Bug"
    },
    "id": "31",
    "label": "Done"
}]
```

Attributes of a transition are values that may be set when performing the transition. Attributes should not be set unless needed or required.

The available attributes and whether or not they are required will vary depending on the type of repository of the collection.

# Payload Transformations

Gateway collections can accept a JSON payload via HTTP, enabling clients to use a REST API to publish artifacts in Tasktop.

Without further configuration, Gateway Collections require a JSON payload that matches the model of the collection.

By configuring a Gateway Collection with an extension, it is possible to accept arbitrarily complex JSON payloads, enabling integration with third party products that integrate with webhooks.

Examples of such third party webhook notifiers include:

- Jenkins Notification Plugin
- Microsoft VisualStudio Web Hooks
- GitHub Webhooks

## Configuring Gateway Collections with Extensions

To configure a Gateway Collection with an Extension, add a payload transformation extension from the Extensions screen, accessible from Settings. Once added, the extension can be referenced from the Gateway Collection screen.

## Authoring Payload Transformation Extensions

Payload transformation extensions are defined by a single function:

```
function transformPayload(payload)
```

The function must return an array of 0 or more JSON objects matching the model of the gateway collection.

Given a model representing build jobs with the following fields:

- `created` - A date signifying the creation date

- `summary` - A brief one-line description
- `status` - A single-select indicating the build status

A simple payload transformation extension could look something like this:

```
function transformPayload(payload) {
  var createdTimestamp = new Date(payload.build.completion_time).toISOString();
  var created = createdTimestamp.substring(0,createdTimestamp.indexOf('T'));
  return [
    {
      'created': created,
      'summary': payload.name + ': '+payload.build.full_url,
      'status': payload.status
    }
  ];
}
```

The example above corresponds to the payload provided by the Jenkins Notification plugin, which provides JSON payloads as follows:

```
{
        "name": "Robot Lawnmower",
        "url": "job/Robot%20Lawnmower/",
        "build":
        {
                "full_url": "http://build.example.com:8081/job/Robot%20Lawnmower/4/",
                "number": 4.0,
                "phase": "COMPLETED",
                "status": "FAILURE",
                "url": "job/Robot%20Lawnmower/4/",
                "scm":
                {

                },
                "causes":
                [
                        "Started by user admin"
                ],
                "duration_string": "9 ms",
                "completion_time": 1.476313762942E12,
                "failing_since_build":
                {
                        "full_url": "http://build.example.com:8081/job/Robot%20Lawnmower/1/",
                        "number": 1.0,
                        "change_set":
                        [

                        ],
                        "completion_time": 1.47631304791E12,
                        "failing_since_time": "11 min"
                }
        }
}
```

Ignoring Webhook Payloads

For cases where the gateway collection is called and no corresponding action should be performed, the extension should return a 0-length array:

```
function transformPayload(payload) {
  ...

  if (nothingToDo) {
     return [];
  }
  ...
}
```

## Creating Multiple Artifacts From A Single Webhook Payload

There may be cases when multiple artifacts should be created from a single webhook payload depending on the use case. For example, a GitHub PushEvent can contain multiple commits. To link each commit to an artifact separately, a payload transformation extension would be used as follows:

```
function transformPayload(payload) {
  var gatewayPayloads = [];
  for each (var commit in payload.commits) {
     gatewayPayloads.push(createCommitPayload(commit));
  }
  return gatewayPayloads;
}
```

## Query Parameters and HTTP headers (optional)

Payload transformations can take two additional parameters — query parameters and HTTP headers:

- **Query Parameters**: Provides the query parameters sent with the payload as a JavaScript object with property names corresponding to parameter names and values as arrays of values of the corresponding query parameter
- **HTTP Headers**: Provides the HTTP request headers sent with the payload as a JavaScript object with property names corresponding to HTTP header names and values as arrays of values of the corresponding HTTP header

This might appear as follows:

```
function transformPayload(payload, parameters, headers)
```

# Custom Data Transformations

In cases where specialized value transformations are needed for use in field mappings, such transformations can be added as **custom data transformation** extensions.

⚠ **Note**: When using a custom data transformation, we recommend **not** mapping an extension that **loses** information from **collection to model**. Because the model is used for change detection, if information is lost, change detection may fail. For example, an extension that transforms a list of links into a single link should not be mapped from **collection to model**, as subsequent change detection would only have a single link to compare rather than the full list. We recommend keeping as much information as possible in the model and mapping any lossy transformations from **model to collection**.

## The Context Object

The context object provides information that the extension can use to determine which transformations are needed.

For a custom data transformation, use the following:

- `context.sourceArtifact:` A JavaScript object representation of the source artifact
  - If you are mapping from model to repository, this will match the structure of the artifact in the model
  - If you are mapping from repository to model, this will match the structure of the artifact in the repository
- `context.targetArtifact:` A JavaScript object representation of the target artifact
  - If you are mapping from model to repository, this will match the structure of the artifact in the repository
  - If you are mapping from repository to model, this will match the structure of the artifact in the model

💡 **Note**: If existing scripts are utilizing `targetRepositoryArtifact` instead of `targetArtifact`, they will continue to work.

If you are creating a custom data transformation extension for test steps, use:

- `context.sourceTestStep` to access the source test step
  - If you are mapping from model to repository, this will match the structure of the artifact in the model
  - If you are mapping from repository to model, this will match the structure of the artifact in the repository
- `context.targetTestStep` to access the target test step
  - If you are mapping from model to repository, this will match the structure of the artifact in the repository
  - If you are mapping from repository to model, this will match the structure of the artifact in the model

*See Tasktop Editions to determine if your edition contains Test Step functionality*

The context parameter also has field properties:

- `context.sourceField`: If processing a single field
- `context.sourceFields`: A list of field objects, if processing more than one field
- `context.targetField`: If processing a single field
- `context.targetFields`: A list of field objects, if processing more than one field

A field object only has two properties: ID, and label:

```
{
    id: "assignee",
    label: "Assignee"
}
```

## Creating a Custom Data Transformation Extension

Custom data transformation extensions are created from the Extensions screen, accessible from Settings. Created extensions can be selected when configuring a field mapping of a collection.

Custom data transformation extensions appear as follows:

```
var inputTypes = 'String';
var outputTypes = 'String';

function transform(context, input) {
        // returns the transformation result
}
```

All custom data transformation extensions must declare their input and output types as shown in the example above. Transformations are only available for a field mapping if the input types and output types match the fields selected in the mapping. In the case of a mapping with multiple source and target fields, the order of the declared input and output types must match the order of the source and target fields.

A simple split-and-trim value custom data transformation extension could look like this:

```
var inputTypes = 'String';
var outputTypes = ['String', 'String'];

function transform(context, input) {
    if (input) {
        var values = input.split('/');
        if (values.length != 2) {
            throw 'Unexpected value ' + input;
        }
        return values.map(function(s) {
            return s.trim();
        });
    }
}
```

## Single Select and Multi Select in Custom Data Transformation Extensions

Single Select and Multi Select values are specified using their labels. Extensions that accept a Single Select as the input type will receive a string containing the option's label. Extensions that specify a Single Select as the output type should return a string containing the option's label.

To specify the empty option, return `undefined` from the extensions instead of a value. Extensions that accept a Multi Select as the input type will receive an array of strings of the option labels. Extensions that specify a Multi Select as the output type should return an array of strings with the option labels or an empty array to specify no options.

If the field has options, field options are available on the field of the context passed into an extension. For example, the options can be accessed with something like this:

```
context.sourceField.options
context.targetField.options
```

## Rich Text Support in Custom Data Transformation Extensions

To perform Rich Text transformations, **Rich Text** must be declared as input or output types of the extension.
A Rich Text input parameter is passed as a valid HTML string.

For Rich Text as output type, the extension is expected to return a valid HTML string.

To escape HTML characters, the following function is provided:

```
html.escape(string)
```

A simple String-to-Rich-Text value transformation could look like this:

```
var inputTypes = 'String';
var outputTypes = 'Rich Text';

function transform(context, input) {
    if (input) {
        return '<pre>' + html.escape(input) + '</pre>';
    }
}
```

## Web Links in Custom Data Transformation Extensions

To perform a web links transformation, web links must be declared as the input or output types of the extension. A web links field consists of a list of web link objects. A web link object consists of a location and other attributes.

The following is an example of a web link output:

```
[
   {
      label: 'Tasktop',
      location: 'http://www.tasktop.com'
   },
   {
      location: 'http://www.alt-tasktop.com'
   }
]
```

💡 **Note**: The label attribute is optional and if specified will be used to populate the label of the web link.

## Relationships in Custom Data Transformation Extensions

Tasktop provides a JavaScript API for working with relationship fields. This API can retrieve, search, and get associated artifacts for artifacts.

Artifact Service API Reference

Artifacts returned from the Artifact API are the raw JSON representation of a Repository's Artifact. These representations may include internal IDs and other fields not mapped to the model. It may be necessary to manually interpret the results of these calls on a per-repository basis to determine the exact information that is returned.

- `artifacts.retrieveArtifact(relationship):Artifact` - Retrieves the artifact for the provided relationship
- `artifacts.listSearchTypes():SearchType[]` - Lists the valid search types for the targeted repository

- `artifacts.getSearchDefinition(searchTypeId):SearchDefinition` - Returns an object with the parameters that are required for the given search type id
- `artifacts.search(searchType, searchDefinition):Relationship[]` - Searches the target repository with the given search type id and search definition, returns a list of relationships which then can be looked up via `artifacts.retrieveArtifact(relationship)` to retrieve the artifact
- `artifacts.getFormattedIdSearchDefinition():SearchDefinition` - Returns an object with the parameters that are required for a formatted ID search
- `artifacts.searchByFormattedId(searchDefinition):Relationship[]` - Searches by formatted ID with the provided search definition and returns a list of relationships which then can be looked up via `artifacts.retrieveArtifact(relationship)` to retrieve the artifact
- `artifacts.toContainer(relationship, summary):Container` - Converts a relationship into a container, summary is optional
- `artifacts.toRelationship(container):Relationship` - Converts a container into a relationship
- `artifacts.getAssociatedRelationship(relationship):Relationship` - Finds the associated relationship for the given relationship. When mapping from model to collection the input value and source artifact relationship field values are from the source repository and must be converted to their associated value to be used in the target system. An exception is thrown if no artifact is found or multiple artifacts are found.
- `artifacts.getAssociatedContainer(container):Container` - Finds the associated container for the given container. When mapping from model to collection the input value and source artifact container link field values are from the source repository and must be converted to their associated value to be used in the target system. An exception is thrown if no artifact is found or multiple artifacts are found.

A sample relationship transformation extension:

```
var inputTypes = 'Relationship';
var outputTypes = 'Relationship';

function transform(context, input) {
        if (input) {
                return findParentFolder(context.sourceArtifact);
        }
        return null;
}

function findParentFolder(artifact) {
        var parent = artifacts.retrieveArtifact(artifact['parent']);
        if (parent['subtype'] === 'Folder') {
                return artifact['parent'];
        } else if (parent['subtype'] === null) {
                return null;
        }
        return findParentFolder(parent);
}
```

Looking at the above extension, we find the parent artifact and if that artifact is a folder we return that as the parent.

```
var inputTypes = 'Relationship';
var outputTypes = 'Relationship';

function transform(context, input) {
```

```
        var searchDefinition = artifacts.getFormattedIdSearchDefinition();

        searchDefinition['formatted-id'] = 'TPA-42';
        var results = artifacts.searchByFormattedId(searchDefinition);
        if (results[0]) {
                return results[0];
        }
        return null;
}
```

The above extensions uses the formatted ID search to find the correct artifact for the link.

The following extension uses a custom search to determine a relationship:

```
var inputTypes = 'Relationship';
var outputTypes = 'Relationship';

function transform(context, input) {
        var searchType = getCustomSearchType();
        var searchDefinition = artifacts.getSearchDefinition(searchType);

        searchDefinition['domain'] = 'DEFAULT';
        searchDefinition['project'] = 'My Project';
        searchDefinition['summary'] = context.sourceArtifact.summary;
        var results = artifacts.search(searchType, searchDefinition);
        if (results[0]) {
                return results[0];
        }
        return null;
}

function getCustomSearchType() {
        var searchTypes = artifacts.listSearchTypes();
        for (var i=0; i<searchTypes.length; i++) {
                if (searchTypes[i] === 'My Custom Search') {
                        return searchTypes[i];
                }
        }
        return i;
}
```

💡 **Note**: The returned search results are limited to a maximum of 1024 entries.

Containers and Relationships

A **Container** can be used as input and output type in a Custom Data Transformation extension. Tasktop provides a JavaScript API for working with container fields.

The following two functions are provided to handle containers:

```
artifacts.toRelationship(container)
```

```
artifacts.toContainer(relationship[, summary])
```

All container objects provide a `summary` property.

- `.toContainer(relationship[, summary])` - Converts a relationship object into a container. The summary is provided as a String and is optional. Is no summary provided, the summary of the related artifact is used. An exception is thrown if the artifact or the summary field of the artifact cannot be found.

- `.toRelationship(container)` - Converts a container into a relationship object to use with the`artifacts.retrieveArtifact(relationship)`
- API or return as result of the extension.

The following extension finds the first parent folder and returns that as the parent container.

```
var inputTypes = 'Relationship';
var outputTypes = 'Container';

function transform(context, input) {
        if (input) {
                var parentRelationship = findParentFolder(context.sourceArtifact);
                  return artifacts.toContainer(parentRelationship);
        }
        return null;
}

function findParentFolder(artifact){
        var parent = artifacts.retrieveArtifact(artifact['parent']);
        if (parent['subtype'] === 'Folder') {
                return artifact['parent'];
        } else if (parent['subtype'] === null) {
                return null;
        }
        return findParentFolder(parent);
}
```

The next extension retrieves the parent of our parent container field and returns it as relationship.

```
var inputTypes = 'Container';
var outputTypes = 'Relationship';

function transform(context, input) {
        if (input) {
                var parentRelationship = artifacts.toRelationship(input);
                var parentArtifact = artifacts.retrieveArtifact(parentRelationship);
                var container = parentArtifact['parent'];
                  return artifacts.toRelationship(container);
        }
        return null;
}
```

💡 **Note**: Only containers based on artifacts are supported.

## Comments in Custom Data Transformation Extensions

Comment extensions can be used to achieve use cases such as:

- Splitting long comments in a source collection into multiple comments in a target collection
- Excluding comments from integration based on some set criteria
- ... and more!

Once saved, the extension can be applied on the Comment Configuration screen.

To create a comment extension, **comments** must be declared as the input or output types of the extension.

💡 **Note**: Comment extensions will only impact new comments as they flow through Tasktop Integration Hub. Existing comments that have already been synchronized will not be impacted.

If you are creating a custom data transformation for comments,

- The **Comments** type is supported as an array of comment objects
- A comment will be a javascript object with field ids as the key
    - For example, a private comment with ID **1** and content **This is a comment** may look like this:

```
{
    "id":"1",
    "is-private":true,
    "comment-content":"<p>This is a comment</p>"
}
```

Here's an example of an extension that replaces user information with a default user in outbound comments:

```
// The following extension can be set on Collection to Model transformation on a collection.
// It replaces user information at a repository's comment to a default user and returns comments that matches
to Hub comment model.
var inputTypes = 'Comments';
var outputTypes = 'Comments';

function transform(context, input) {
    if(input.length >0){
        input.forEach(function(element) {
            replacePeople(element);
        });
    }
     return input;
}

function replacePeople(comment){
    var pattern = /user(\d*)/gi;
    comment['creator']='default';
    comment['work_notes']=comment['work_notes'].replace(pattern,'default');  //replace user information at a
repository's comment contents field, work_notes
    comment['content']=comment['work_notes'];  //assign updated repository's comment content to the Hub's
comment object's comment content field
}
```

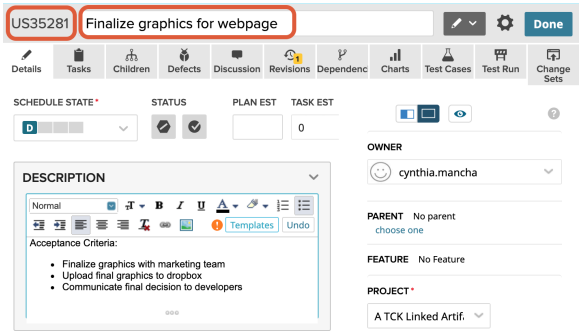
Here's an example of an extension that adds a header to inbound comments with a default user:

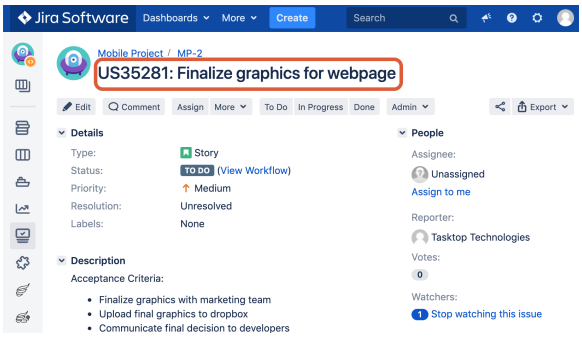
```
// The following extension can be set on Model to Collection transformation on a collection.
// It addes comment header with default user to the given Hub model's comment input.
var inputTypes = 'Comments';
var outputTypes = 'Comments';

function transform(context, input) {
    if(input.length >0){
            addCommentHeader(input[0]);
    }
     return input;
}

function addCommentHeader(comment){
    var headerText = '<p>[Comment from '+'default_user'+']</p>';
    comment['content']=headerText+comment['content'];  //Hub's comment object's comment content field is comment
['content']
}
```

# Concatenation

To concatenate two fields on the source artifact into one field on the target artifact, a custom data transformation extension can be used.

Below, we've outlined how to configure a custom data transformation extension in order to concatenate the Formatted ID and Name from CA Agile Central into the Summary model field. The concatenated values will then flow from the model to the chosen field on the target artifact.




1. Go to the Field Mapping screen for the source (CA Agile Central) collection.
2. If the Summary model field is already mapped in the source collection, delete the mapping.
3. Choose Formatted ID and Name from the left side (repository) dropdown and Summary from the right side (model) dropdown and Press Connect.
4. Make a note of the Type for each of the 2 fields and the order in which they are added. For example, in the below example Formatted ID was added first and is of type **String** and Name was added next and is of type **String**. The Model Field is also of type **String**.

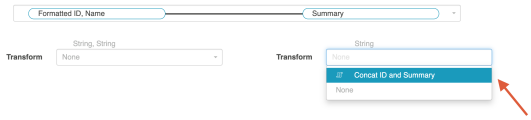

5. Open the Settings in a different tab and go to Extensions > Manage Extensions.
6. Create a new data transformation extension.
7. Give the extension a name and update the input types based on Step 5. In this case we have 2 Inputs of types **String** and **String**. Update the input types as follows:

```
var inputTypes = ['String', 'String'];
```

   a. 💡 **Note**: This will take the Formatted ID as the 1st parameter and Name as the 2nd parame ter.

8. Update the output types based on Step 5. In this example, we have 1 output of type **String**. Update output types as follows:
    a. 
```
var outputTypes = 'String';
```
9. In the body of the function, use the following statement to concatenate:
    a. 
```
return 'ID: ' + input[0] + ' :: '+input[1];
```
10. Here's an example of the full script:
    a. 
```
var inputTypes = ['String', 'String'];
var outputTypes = 'String';


function transform(context, input) {
   // returns the result of the transformation
   return 'ID: ' + input[0] + ' :: '+input[1];
}
```
11. Save and go back to the source collection.
12. Configure the Summary mapping from Step 4:


13. You will now see the extension you created as an option for the transform on the right (model) side. Choose this extension and click **Save** and **Done**.




14. In your target collection, simply map the Summary model field to your chosen field on the target artifact (i.e., Summary, Name, Title, etc).
    a. 


This will concatenate the 2 fields (ID, Name) on the source artifact to a single Summary field on the target artifact.

# Person Reconciliation

Integrations that create or update artifacts often need to deal with differences between the representation of persons in different systems.

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 (person-username) from source to username (person-username) on target
- Username (person-username) from source to ID (person-id) on target
- ID (person-id) from source to username (person-username) on target

- ID (person-id) from source to ID (person-ID) on target
- Email (person-email) from source to email (person-email) from target

Please review the Connector Docs to determine which fields are available for your specific repository. If a field (i.e., person-username) is not available, Tasktop will simply skip that step.

This strategy should cover most use cases. However, if needed, you can also configure a custom Person Reconciliation Extension to match **person** fields from one repository to another.

## Configuring Person Reconciliation with Extensions

A person reconciliation extension can be created from the Extensions screen, accessible from Settings. Created extensions are selected in the Person Reconciliation section of the Collection screen. In most cases it makes sense to have one extension per repository, since each repository will have different requirements for mapping persons to and from the repository. Person reconciliation extensions apply to all person fields of an artifact, including person fields in comments and attachments.

## The Context Object

The context object provides information that the extension can use to determine how person reconciliation should be handled.

For a custom data transformation, use the following:

- `context:` A context object that provides information that the extension can use to determine which transformations are needed
- `context.sourceArtifact:` A JavaScript object representation of the source artifact
  - If you are mapping from model to repository, this will match the structure of the artifact in the model
  - If you are mapping from repository to model, this will match the structure of the artifact in the repository
- `context.targetArtifact*:` A JavaScript object representation of the target artifact
  - If you are mapping from model to repository, this will match the structure of the artifact in the repository
  - If you are mapping from repository to model, this will match the structure of the artifact in the model

If you are creating a custom data transformation extension for test steps, use:

- `context.sourceTestStep` to access the source test step
  - If you are mapping from model to repository, this will match the structure of the artifact in the model
  - If you are mapping from repository to model, this will match the structure of the artifact in the repository
- `context.targetTestStep` to access the target test step
  - If you are mapping from model to repository, this will match the structure of the artifact in the repository

- If you are mapping from repository to model, this will match the structure of the artifact in the model

*See the Tasktop Editions to determine if your edition contains Test Step functionality*

The context parameter also has field properties:

- `context.sourceField`: If processing a single field
- `context.sourceFields`: A list of field objects, if processing more than one field
- `context.targetField`: If processing a single field
- `context.targetFields:` A list of field objects, if processing more than one field

A field object only has two properties: ID, and label:

```
{
    id: "assignee",
    label: "Assignee"
}
```

## Authoring Person Reconciliation Extensions

Person reconciliation extensions are defined by two functions:

```
mapPersonFromRepository(repositoryPerson, unresolvedPerson)
```

```
mapPersonToRepository(modelPerson)
```

Both functions are expected to return a string value corresponding to the user id of the person. Returning `undefined` sets the person field to empty. In the case where a user cannot be mapped and having the field empty is not an option, throw an exception as follows:

```
if (noMatchFoundCondition) {
    throw 'some descriptive message';
}
```

Such errors will cause processing of an artifact to result in an error with error code CCRRTT-17011E which will display under the Activity screen.

mapPersonFromRepository(repositoryPerson, unresolvedPerson)

`mapPersonFromRepository` is used to create a model representation of a person from a repository representation of a person, which occurs whenever a person is copied from a repository artifact to a model artifact. The return value of this function is used as the id of the person in the model artifact.

Two parameters are passed to the `mapPersonFromRepository` function:

- `repositoryPerson` - An object representing the person corresponding to the repository representation
- `unresolvedPerson` - This parameter contains whatever information may be available about the person from the repository. It contains information only if `repositoryPerson` does not.

An example `repositoryPerson` from Jira On-prem looks like:

```
{
        "person-id": "userA",
        "person-email": "userA@test.tasktop.com",
        "person-display-name":"User A",
        "active": true
}
```

An example `unresolvedPerson` from Jira On-prem might look like:

```
{
        "person-id": "userA",
        "person-email": "userA@test.tasktop.com"
}
```

mapPersonToRepository(modelPerson)

`mapPersonToRepository` is used to create a repository representation of a person from a model representation of a person, which occurs whenever a person is copied from a model artifact to a repository artifact. The return value of this function is used to lookup the corresponding person in the repository.

A single parameter is passed to the `mapPersonToRepository` function:

- `modelPerson` - An object representing the person corresponding to the model representation

A `modelPerson` always has the following properties:

```
{
        "id": "userId",
        "display-name": "Jane Smith"
}
```

💡 **Note**: `Display-name` could be empty.

Simple Person Reconciliation Example

A simple person reconciliation mapping extension could look like this:

```
function mapPersonFromRepository(repositoryPerson, unresolvedPerson, context) {
    if (repositoryPerson) {
        return repositoryPerson.id;
    }
}

function mapPersonToRepository(modelPerson, context) {
    if (modelPerson) {
        try {
            var person = persons.searchPerson('id', modelPerson.id);
            console.log("found match " + person.id);
            return person.id;
        } catch (e) {
            console.log("no match found mapping to " + context.targetField.id);
            if (context.targetField.id === "assignee") {
                return "default-assignee";
            } else if (context.targetField.id === "reporter") {
                return "default-reporter";
            } else if (context.targetField.id === "comments") {
                return "default-commenter";
            }
```

```
        }
    }
}
```

The SimplePersonReconciliation script is a simple script which makes use of dictionary concept in Javascript to map key and values.

Scenario 1: Using E-mail

Consider an example where Repository 1 has email **john.s@email.com** and Repository 2 has email **john.smith@email.com** and the display names and ID's don't match. Assume that the integration has one-way person flow from Repository 1 (john.s@email.com) to Repository 2 (john.smith@email.com).

In that case, we would edit the var mapping on the `mapPersonToRepository()` function so that the incoming value checks the dictionary (key) and returns a valid email (value) for the repository.

In this example, we would edit the `var mapping = { 'john.s@email.com' : 'john.smith@email.com'}` in the `mapPersonToRepository()` function.

If the integration has two-way person flow, we must also edit the `mapPersonFromRepository()` function. The `mapPersonFromRepository()` function will show the e-mail addresses in the opposite order - i.e. `var mapping = {'john.smith@email.com' : 'john.s@email.com'}`. For two-way integrations, the person reconciliation extension must be added to both the source collection and the target collection.

Scenario 2: Using ID

If the source repository does not provide an e-mail, we can use the Simple Person Reconciliation script above to match person ID to person email.

For example, if Repository 1 has user id "JohnSmith" and the matching user in Repository 2 is "john.smith@email.com," then we should edit the script at `var mapping = { JohnSmith: 'john.smith@email.com'}`.

If the integration has two-way person flow, we will also need to edit the `mapPersonFromRepository()` as outlined in Scenario 1. We must also remember to edit the extension in `var result` as `modelPerson[person-id]` for scenarios where we are using ID instead of email. The edit must be done on both the `mapPersonFromRepository()` and `mapPersonFromRepository()` functions.

Selecting a Default Person when No Match is Found

Below is a script which uses the context to select a default person when no match is found:

```
function mapPersonToRepository(modelPerson, context) {
        var person;
        try {
                person = persons.searchPerson("email", modelPerson.id);
        } catch (e) {
                // no matching person found
                // select a default person by team
                if (context.sourceArtifact["team"] === "Team A") {
                        person = persons.searchPerson("email", "team.a.lead@company.net");
                } else if (context.sourceArtifact["team"] === "Team B") {
                        person = persons.searchPerson("email", "team.b.lead@company.net");
                }
```

```
        }
        // return a match if found
        if (person) {
                return person["person-id"];
        }
}
```

Person Reconciliation Extension Javascript API

Tasktop provides a JavaScript API for working with persons in a person reconciliation extension. This API includes two functions:

- `persons.listPersonSearchFields():Object\` - Allows for the discovery of the searchable fields on a person. Not all fields from a person are searchable and vary between connectors.
- `persons.searchPerson(fieldId, fieldValue):Person\` - Used to search for person in a repository. This person can then be used to return the correct ID for a user in a repository. `persons.searchPerson(fieldId, fieldValue)` will find exactly one person and will throw a `PersonNotFoundException` if no match is found or `TooManyPersonsFoundException` if more than one person is found. These exceptions can be caught and handled by the extension.

Artifacts returned from the artifact API are the raw JSON representation of a repository's artifact. These representations may include internal IDs and other fields not mapped to the model. It may be necessary to manually interpret the results of these calls on a per-repository basis to determine the exact information that is returned.

Below is a person reconciliation extension that will take the id of a model person, retrieve the user by username and return the exact ID from the repository. This is helpful for systems where the person's ID is a number or some other non-human readable value.

```
function mapPersonFromRepository(repositoryPerson, unresolvedPerson) {
    return repositoryPerson['Username'];
}

function mapPersonToRepository(modelPerson) {
    // persons.listPersonSearchFields(); determines the fields usable by .searchPerson(...)
    var repositoryPerson = persons.searchPerson('Username', modelPerson['id']);
    return repositoryPerson['ID'];
}
```

## SearchPerson Example Script

Below is an example SearchPerson script. `Persons.searchperson (fieldId, fieldValue)` is used to search for a person in a repository using the two parameters: `fieldId` and `fieldValue`.

This person can then be used to return the matching ID of a user in that repository.

`Persons.searchPerson(fieldId, fieldValue)` will find exactly one person and will throw a `PersonNotFoundException` if no match is found or `TooManyPersonsFoundException` if more than one person is found. These exceptions can be caught and handled by the extension. `searchPerson()` is a native Tasktop call, which means it is functionality that is unique to Tasktop.

```
function mapPersonFromRepository(repositoryPerson) {
            ...
```

```
}

function mapPersonToRepository(modelPerson) {
            if (!modelPerson){
                   console.log('incoming model person is empty')
                   return undefined
            }

            console.log('modelPerson = ' + modelPerson['id']);

   var repoPerson = persons.searchPerson('person-username', modelPerson['id']);

   console.log('repoPerson = ' + repoPerson['id']);
        return repoPerson['id'];
}
```

Scenario 1: Mismatched E-mails

Consider an example where Repository 1 has email **john.s@email.com** and Repository 2 has email **john.smith@email.com**. The `persons.searchPerson(fieldId, fieldValue)` can be used to search the repository for matching person values.

Assume that the integration has one-way person flow from Repository 1 (john.s@email.com ) to Repository 2 (john.smith@email.com ). In this case, the `mapPersonToRepository()` function should be edited and the incoming values matched by ID. A search persons call based on incoming username is made and then the matching user object is retrieved.

Scenario 2: Returning a Default ID as a Value

```
function mapPersonFromRepository(repositoryPerson) {
        return repositoryPerson['email'];
}

function mapPersonToRepository(modelPerson) {
            var defaultUserId = 'SOMEVALUEHERE'

            console.log(persons.listSearchFields())
            try{
                    var person = persons.searchPerson('email', modelPerson.id);
                    if(person != null) {
                            return person['person-username'];
                    }
        } catch(e){
                console.log(e)
        }
        console.log('Falling back to default person')
        return defaultUserId
}
```

The above script allows us to search for persons in the repository based on an incoming email value. In cases where a corresponding person is not found in the repository, Tasktop will return the `defaultUserId`. To return a default user ID, assign a default value (a user ID) to the `var defaultUserId`.

PersonListSearchFields

`Persons.listPersonSearchFields()` allows for the discovery of the searchable fields on a person. Not all fields from a person are searchable and vary between connectors.

```
function mapPersonFromRepository(repositoryPerson) {
        return repositoryPerson['email'];
}

function mapPersonToRepository(modelPerson) {
```

```
            console.log(JSON.stringify(persons.listPersonSearchFields()))
        var person = persons.searchPerson('person-email', modelPerson.id);
        return person['person-id'];
}
```

For example, when using the above Person Reconciliation script/extension on the Jira side in a Jira-Micro Focus (HPE) integration, the `console.log(JSON.stringify(persons.listPersonSearchFields()))` line will give you a list of the searchable fields.

In our demo, we got the following values:

```
Person listSearch Fields: ["person-username","person-email","person-id","person-display-name"]
```

You can then use one of those available values as part of the `persons.searchPerson() script`. In the example scripts shown above, we make use of `person-id`.

# Using LDAP or Active Directory

LDAP (Lightweight Directory Access Protocol) and Active Directory can be used to lookup information required to map persons from one system to another. Tasktop provides a JavaScript API for accessing LDAP and Active Directory as follows:

```
function mapPersonToRepository(modelPerson) {
        ldap.connect('ldap://subdomain.mycompany.com', 'cn=admin,dc=example,dc=mycompany,dc=com', 'mypassword');
        var results = ldap.search('dc=example,dc=mycompany,dc=com', 'cn='+ldap.escape(modelPerson['id']))
        if (results.length == 0) {
           throw 'no person found with id='+modelPerson['id'];
        }
        return results[0]['sn'];
}
```

Looking at the example above, three steps are involved:

1. Establishing a connection
2. Finding the appropriate entries using a search
3. Returning a value from the search results

The same approach is used for both LDAP and Active Directory.

The Tasktop JavaScript LDAP API is described as follows:

- `ldap` - The globally-visible object providing the LDAP API
- `ldap.connect(connectionUrl, principal, password):void` - A means of establishing a connection with a connection URL, user principal and password
- `ldap.search(base, query, fields):Map[]` - A means of searching providing a base name of the context to search, a search query, and an optional list of fields to provide in the search results
- `ldap.escape(value):String` - A means of escaping string literals to use in LDAP search queries or distinguished names

There is no need to close an LDAP connection; LDAP connections are managed implicitly by Tasktop.

Artifacts returned from the Artifact API are the raw JSON representation of a repository's artifact. These representations may include internal IDs and other fields not mapped to the model. It may be necessary to manually interpret the results of these calls on a per-repository basis to determine the exact information that is returned.

## Accessing Web Resources

Extensions may access resources using HTTP. For example, extensions may access a REST API which could provide data necessary for the extension.

Tasktop provides a fluent JavaScript API for making HTTP requests, inspired by the Java 9 HTTP client API. The API is used as follows:

```
var response = httpClient.request()
        .uri('http://example.com/my/rest/api')
        .parameter('first-param','first-value')
        .parameter('second-param','second-value')
        .header('my-special-header','header-value')
        .GET().response()

if (response.statusCode() == 200) {
        var responseJson = JSON.parse(response.content());
        // do something with response data
}
```

## HTTP Client API Reference

- `httpClient` - The globally-visible object providing the HTTP client API
- `httpClient.request():RequestBuilder` - Provides a RequestBuilder object
- `RequestBuilder.uri(uriString):RequestBuilder` - Specifies the URI of the request
- `RequestBuilder.parameter(key,value):RequestBuilder` - Adds a query parameter to the request with the given key and value
- `RequestBuilder.header(key,value):RequestBuilder` - Adds an HTTP header value to the request with the given key and value
- `RequestBuilder.GET():Request` - Creates a Request object for an HTTP GET request
- `Request.response():Response` - Creates a Response object with the result of the HTTP request
- `Response.statusCode():int` - Provides the HTTP status code of the response
- `Response.content():String` - Provides the body of the HTTP response as a string
- `Response.headers():Map` - Provides the HTTP response headers as a JavaScript object with property names corresponding to HTTP header names, and values as arrays of values of the corresponding HTTP header
- `RequestBuilder.proxy(hostname, port[, username, password]): RequestBuilder` - Specifies an HTTP proxy to be used for this request

Artifacts returned from the Artifact API are the raw JSON representation of a Repository's Artifact. These representations may include internal IDs and other fields not mapped to the model. It may be necessary to manually interpret the results of these calls on a per-repository basis to determine the exact information that is returned.

Example extension `Response.headers()` return value:

```
{
        "Transfer-Encoding": [
                "chunked"
        ],
        "Server": [
                "Jetty(9.2.13.v20150730)"
        ],
        "Vary": [
                "Accept-Encoding, User-Agent"
        ],
        "Content-Type": [
                "application/json;charset=UTF-8"
        ]
}
```

## Using an HTTP Proxy Server

Extensions can specify an HTTP proxy server with the following API:

```
var response = httpClient
    .request()
    .proxy("myproxy.mycompany.com", 3128)
    .uri('https://www.example.com')
    .GET()
    .response();
```

To use a proxy server with BASIC proxy authentication, credentials can be specified as shown below assuming username and password are strings:

```
var response = httpClient
    .request()
    .proxy("myproxy.mycompany.com", 3128, username, password)
    .uri('https://www.example.com')
    .GET()
    .response();
```

## Good to Know

- All communication to the proxy server uses HTTP, not HTTPS, so even if an HTTPS connection to the target server is tunneled through the proxy, it is important that the connection to the proxy server is through a trusted network if sending proxy credentials.
- We recommend storing the proxy password in a confidential key-value store and not hard coding it in the extension because extensions are stored unencrypted in Hub's operational database.

## Causing Extensions to Complete With An Error

There are occasions where extensions should complete with an error. In such cases, simply use the JavaScript `throw` keyword as follows:

```
if (somethingUnexpected) throw 'some descriptive message'
```

Such errors will cause processing of an artifact to result in an error with error code CCRRTT-17011E which will display on the Activity screen.

# Troubleshooting Extensions

Extension troubleshooting usually involves trial and error. To make the troubleshooting process easier, a global logging function is exposed as follows:

`console.log(message)`

`console.log` takes a single argument which is converted to a string.

For example:

```
function transitionArtifact(context,transitions) {
    if (someUnexpectedCondition) {
        console.log('source artifact: '+JSON.stringify(context.sourceArtifact));
        console.log('target artifact: '+JSON.stringify(context.targetRepositoryArtifact));
        console.log('transitions: '+JSON.stringify(transitions));
        throw 'message describing that something bad happened';
    }
}
```

The output of `console.log` goes to the Tasktop log file at `logs/extensions.log`

# Extensions and State

Extensions should not rely on declared variables to retain state between invocations. Doing so is not supported and has undefined behavior.
For example:

// This is not supported:

```
var myGlobalState = // some state


function someFunction() {
   if (myGlobalState == someValue) {
     ...
   }
}
```

# Accessing Object Properties

There are two ways to access object properties:

### Dot Notation

You can use the dot notation if the property name only contains alpha-numeric and characters that are allowed in JavaScript variables such as '$' or '_'.

For example:

```
person.email
```

### Bracket Notation

You must use the bracket notation if the property name contains characters that are not allowed in JavaScript variables such as a hyphen.

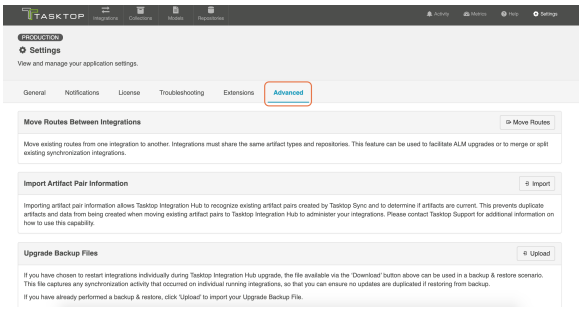
For example:

```
person['id']
```

# Advanced (Settings)

## Introduction

Advanced (Settings) can be accessed by clicking the **Advanced** tab on the **Settings** screen.




Under **Advanced**, you can access:

- Flow External Workflow Changes
- Move Routes Between Integrations
- Import Artifact Pair Information
- Upgrade Backup Files

Under General (Settings), you can access:

- Configuration
- Master Password Configuration
- Storage Settings

Under Notifications, you can access:

- Email Notifications

Under License, you can access:

- License

Under Troubleshooting, you can access:

- Logging

Under Extensions, you can access:

- Extensions
- Key-Value Stores

## Move Routes Between Integrations

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. Since existing projects are moved to a completely new ALM instance with a different URL, users must create a new repository connection, collection(s), and integration(s) in Tasktop. Once the new integration is created, existing routes must be migrated to prevent the risk of duplicate artifacts. This feature will allow users to easily migrate routes from an existing integration to a new one.

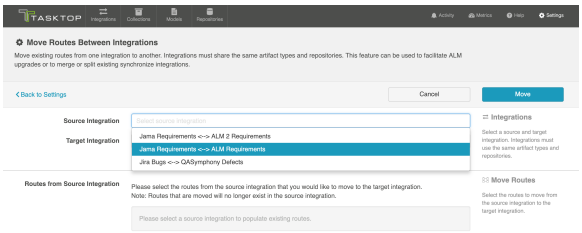
To move routes from one integration to another, they must both:

- be synchronize integrations
- use the same artifact types
- use the same repository connections (except for Micro Focus/HPE ALM connections used in an upgrade scenario)

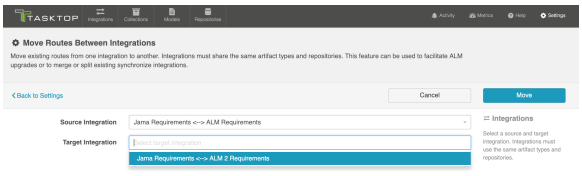
💡 We recommend stopping both integrations before moving routes so that you can review your mappings and configuration before running.

To use this feature click **Move Routes**.

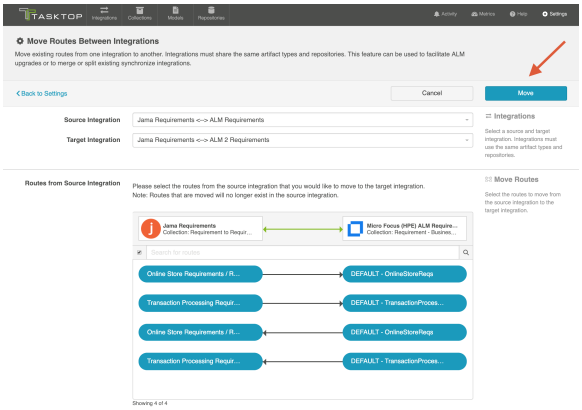

Select your source integration:




Then select your target integration:




Select the routes from the source integration that you'd like to move to the target integration. Once moved, they will no longer exist in the source integration. Click **Move** in the upper right corner.




Review the pop-up message and if approved, click **I understand...** and **Move**. This process may take some time. Progress can be tracked on the Background Jobs tab of the Activity Screen.

Once the move is complete, review your integration configuration, field mappings, etc, before clicking **run** on the target integration.

# Import Artifact Pair Information

Importing artifact pairs allows Tasktop Integration Hub to know about existing artifact pairs that were created by Tasktop Sync. This prevents duplicate artifacts from being created when you switch from using Tasktop Sync to Tasktop Integration Hub to administer your integrations.

Please contact Tasktop Support for additional information on how to use this capability.

# Upgrade Backup Files

***This feature is not applicable to Tasktop Cloud and is only available when upgrading from Tasktop Integration Hub versions 20.1 and later.***

Upgrading backup files enables you to download and upload artifact data in cases where integrations were resumed individually during an upgrade. The downloaded data corresponds to artifacts that were modified when migrations were still running. These files capture any synchronization activity that occurred on individually running integrations, so that you can ensure no updates are duplicated if restoring from backup.

Learn more about utilizing this capability here.

# 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.

# Supported Repository Versions

## Tasktop Integration Hub 21.3 (July 20, 2021)

Please check the Repository Versions: End-of-Life Dates and Extended Support page for more information on End-of-Life dates and extended support. 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<br><br>(Tasktop 21.3) |
|---|---|---|
| aras INNOVATOR | Aras Innovator | 11.0 SP15 |
| Jira Core | Atlassian Jira Core | 7.10, 7.11, 7.12, 7.13, 8.0, 8.1, 8.2, 8.3, 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<br><br>Current On Demand (Cloud) Version |
| Jira Software | Atlassian Jira Software | 7.10, 7.11, 7.12, 7.13, 8.0, 8.1, 8.2, 8.3, 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<br><br>Current On Demand (Cloud) Version |
| | Atlassian Jira Service Management | 3.10, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 4.0, 4.1, 4.2, 4.3, 4.4 |

| | | |
|---|---|---|
|  | | |
|  | Atlassian Jira Align | Current On Demand (Cloud) Version |
|  | Blueprint | 9.0, 9.1, 10.0, 10.1, 10.2, 10.3, 11.0, 11.1<br><br>4.0, 4.1, 4.2, 5.0, 5.1 (Storyteller)<br><br>Current On Demand (Cloud) Version<br><br>**Note**: With the release of Blueprint version 11.2, Blueprint Storyteller has been rebranded as Blueprint. Blueprint Storyteller versions 4.0 - 5.1 remain supported. |
|  | BMC Remedy | 9.1.00, 9.1.02, 9.1.03, 9.1.04, 18.08, 18.05, 19.02, 19.08, 19.11, 20.02 |
| | CA PPM | 15.5, 15.5.1, 15.6, 15.7, 15.7.1, 15.8, 15.8.1, 15.9, 15.9.1<br><br>Current On Demand (Cloud) Version |
| | CA Agile Central | 2018.1, 2.0 |

| | | |
|---|---|---|
|  | (Rally) | Current On Demand (Cloud) Version |
|  | Cherwell Service Management<br><br>(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 |
|  | Git | All<br><br>*Note: If using a supported Git Hosting Service, the version of the service used does not impact functionality. It is used to determine commit location.<br><br>⚠️ Git connector is not available on Tasktop Cloud |
| | GitHub Issues | Enterprise 11.10.343, 2.3 and higher, |

| | | |
|---|---|---|
|  | | Current On Demand (Cloud) Version |
|  | GitLab Issues | Enterprise and Community Edition: 10.0, 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.8, 11.0, 11.1, 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 11.8, 11.9, 11.10, 11.11, 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, 14.0<br><br>Current On Demand (Cloud) Version |
|  | IBM Rational ClearQuest | 9.0, 9.0.1 |
| | IBM Engineering Requirements Management DOORS Family | 9.5, 9.5.2, 9.6, 9.6.1, 9.7, 9.7.1<br><br>⚠ IBM DOORS connector is not available on Tasktop Cloud |
| | IBM Engineering Requirements Management DOORS Next | 6.0, 6.0.1, 6.0.2, 6.0.3 iFix 005 and later, 6.0.4, 6.0.5, 6.0.6, 7.0, 7.0.1, 7.0.2 |
| | IBM Engineering Test Management | 6.0, 6.0.1, 6.0.2, 6.0.3, 6.0.4, 6.0.5, 6.0.6, 7.0, 7.0.1, 7.0.2 |
| | IBM Engineering Workflow Management | 6.0, 6.0.1, 6.0.2, 6.0.3, 6.0.4, 6.0.5, 6.0.6, 7.0, 7.0.1, 7.0.2 |
| | Jama Connect | |

| | | |
|---|---|---|
|  | | 8.36, 8.38, 8.39, 8.42, 8.49, 8.56<br><br>Current On Demand (Cloud) Version |
|  | Micro Focus ALM /Quality Center | On Premise and SaaS versions: 12.5, 12.53, 12.55, 12.60<br><br>14.00-SaaS (Patch 0), 14.01 (SaaS), 15.0, 15.0.46, 15.5 |
| | Micro Focus ALM Octane | 12.53 (inclusive only of 12.53.20 and higher), 12.55, 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<br><br>Current On Demand (Cloud) Version |
| | Micro Focus Dimensions RM | 12.6, 12.6.1, 12.6.2, 12.7, 12.8 |
| | Micro Focus PPM | 9.4, 9.41, 9.42, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.62, 9.64 |
| | Micro Focus Solutions Business Manager | 11.3, 11.4, 11.4.1, 11.5, 11.7 |
| | Microsoft Azure DevOps Server | 2013, 2013.2, 2013.3, 2013.4, 2015, 2015.1, 2015.2, 2015.3, 2017, 2017.1, 2017.2, 2017.3, 2017.3.1 2018, 2018.1, 2018.2, 2018.3, 2019 RC1, 2019, 2020<br><br>Current On Demand (Cloud) Version |
| | | |

| | | |
|---|---|---|
|  | Microsoft Azure DevOps Services | Current On Demand (Cloud) Version*<br><br>*Please note limitations in Connector Documentation |
| | Microsoft Project Server | 2013 SP1, 2016*, 2019*, Project Online*<br><br>*Please note limitations in Connector Documentation |
| | 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 |
|  | Pivotal Tracker | Current On Demand (Cloud) Version |
| | Planview Enterprise One | 16 On Demand, 17 On Demand, 18 On Demand |
| | Planview LeanKit | Current On Demand (Cloud) Version |
| | Planview PPM Pro | Current On Demand |

| | | |
|---|---|---|
|  | | (Cloud) Version |
|  | Polarion ALM | 19, 19.1, 19.2, 19.3 |
|  | PTC Windchill | 11.1, 12.0, 12.0.1 |
| | PTC Windchill RV&S | 11.0, 11.1, 11.2, 12.0, 12.1, 12.2, 12.3 |
|  | QASymphony qTest Manager | 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<br><br>Current On Demand (Cloud) Version |
|  | Salesforce: Sales Cloud, Service Cloud, Marketing Cloud | Current On Demand (Cloud) Version |
| | ServiceNow:<br><br>IT Service Management, | Madrid On Demand, New York On Demand, Orlando On Demand, Paris On Demand, Quebec On Demand |

| | | |
|---|---|---|
| servicenow | IT Business Management (Agile Development/SDLC, PPM) | |
| | ServiceNow Express | Current On Demand (Cloud) Version |
| SMARTBEAR | 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, 12.11, 12.12, 12.13, 12.14, 12.20, 12.21, 12.31<br><br>Current On Demand (Cloud) Version |
| Pro Cloud Server | Sparx Systems Pro Cloud Server | 2.0, 2.1.18, 2.1.19, 2.1.20, 2.1.21, 3.0, 4.0<br><br>*Premium Editions only |
| targetprocess SEE. CHANGE. | Targetprocess | Current On Demand (Cloud) Version |
| TRICENTIS | Tricentis Tosca | 11.0, 11.1, 11.2, 11.3, 12.0, 12.1, 12.2, 12.3, 13.0, 13.1, 13.2, 13.3, 13.4, 14.0, 14.1, 14.2 |
| | Trello | Current On Demand (Cloud) Version - Business Class Edition |

| | | |
|---|---|---|
|  | | |
|  | VersionOne | Enterprise and Ultimate: 19.2 (Summer 2019), 19.3 (Fall 2019), 20.0 (Winter 2020), 20.1 (Spring 2020), 20.2 (Summer 2020), 20.3 (Fall 2020), 21.0, 21.1<br><br>Current On Demand (Cloud) Version |
|  | Whitehat Sentinel | Current On Demand (Cloud) Version |
|  | XebiaLabs XL Release | 8.0, 8.2, 8.5 |
|  | Zendesk | Current On Demand (Cloud) Version |
|  | Zephyr for Jira | 3.2.1, 3.2.2, 3.3, 3.3.2, 3.4, 3.5, 3.6, 4.0, 5.x<br><br>Current On Demand (Cloud) Version |

# Repository Versions: End-of-Life Dates and Extended Support

## Tasktop Integration Hub 21.3 (July 20, 2021)

💡 The following repository versions are available for Tasktop's extended support until the End-of-Life date indicated. If you are interested in extended support, please reach out to your Tasktop contact.

### Repository Versions:

| Release | End-of-Life Date |
|---|---|
| 8.3 | 22 Jul 2022 |
| 8.4 | 09 Sep 2022 |
| 8.5 | 21 Oct 2022 |
| 8.6 | 17 Dec 2022 |
| 8.7 | 03 Feb 2023 |
| 8.8 | 19 Mar 2023 |
| 8.9 | 20 May 2023 |
| 8.10 | 23 Jun 2023 |
| 8.11 | 15 Jul 2023 |
| 8.12 | 26 Aug 2023 |
| 8.13 | 08 Oct 2023 |
| 8.14 | 23 Nov 2023 |

## Supported Repository Versions

💡 The following repository versions are currently supported in Tasktop Hub version 21.3.

| | Repository | General Support<br><br>(Tasktop 21.3) |
|---|---|---|
| | Aras Innovator | 11.0 SP15 |
| Jira Core | Atlassian Jira Core | 7.10, 7.11, 7.12, 7.13, 8.0, 8.1, 8.2, 8.3, 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<br><br>Current On Demand (Cloud) Version |
| Jira Software | Atlassian Jira Software | 7.10, 7.11, 7.12, 7.13, 8.0, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11, 8.12, 8.13, |

| | |
|---|---|
| 8.15 | 02 Feb 2024 |
| 8.16 | 23 Mar 2024 |

| Release | End-of-Life Date |
|---|---|
| 8.3 | 22 Jul 2022 |
| 8.4 | 09 Sep 2022 |
| 8.5 | 21 Oct 2022 |
| 8.6 | 17 Dec 2022 |
| 8.7 | 03 Feb 2023 |
| 8.8 | 19 Mar 2023 |
| 8.9 | 20 May 2023 |
| 8.10 | 23 Jun 2023 |
| 8.11 | 15 Jul 2023 |
| 8.12 | 26 Aug 2023 |
| 8.13 | 08 Oct 2023 |
| 8.14 | 23 Nov 2023 |
| 8.15 | 02 Feb 2024 |
| 8.16 | 23 Mar 2024 |

| Release | End-of-Life Date |
|---|---|
| 4.3 | 22 Jul 2022 |
| 4.4 | 09 Sep 2022 |

| Release | End-of-Life Date |
|---|---|
| 9.1.03 | 08 Jun 2023 |
| 9.1.04 | 05 Dec 2023 |
| 18.05 | 31 May 2024 |
| 18.08 | 31 Aug 2024 |
| | |

| | | |
|---|---|---|
| | | 8.14, 8.15, 8.16, 8.17<br><br>Current On Demand (Cloud) Version |
| | Atlassian Jira Service Management | 3.10, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 4.0, 4.1, 4.2, 4.3, 4.4 |
| | Atlassian Jira Align | Current On Demand (Cloud) Version |
| | Blueprint | 9.0, 9.1, 10.0, 10.1, 10.2, 10.3, 11.0, 11.1<br><br>4.0, 4.1, 4.2, 5.0, 5.1 (Storyteller)<br><br>Current On Demand (Cloud) Version<br><br>**Note**: With the release of Blueprint version 11.2, Blueprint Storyteller has been rebranded as Blueprint. Blueprint Storyteller versions 4.0 - 5.1 remain supported. |

| | |
|---|---|
| 19.02 | 28 Feb 2025 |
| 19.08 | 22 Aug 2025 |
| 20.02 | 21 Feb 2026 |

| Release | End-of-Life Date |
|---|---|
| 15.7 | 30 Sep 2022 |
| 15.7.1 | 30 Sep 2022 |
| 15.8 | 31 Mar 2023 |
| 15.9 | 30 Nov 2024 |
| 15.9.1 | 30 Nov 2024 |

| Release | End-of-Life Date |
|---|---|
| 11.0 | 22 Jun 2022 |
| 11.1 | 22 Jun 2022 |
| 11.2 | 22 Jun 2022 |
| 11.3 | 22 Jun 2022 |
| 11.4 | 22 Jun 2022 |
| 11.5 | 22 Jun 2022 |
| 11.6 | 22 Jun 2022 |
| 11.7 | 22 Jun 2022 |
| 11.8 | 22 Jun 2022 |
| 11.9 | 22 Jun 2022 |
| 11.10 | 22 Jun 2022 |
| 11.11 | 22 Jun 2022 |

| Release | End-of-Life Date |
|---|---|
| 12.60 | 31 Aug 2023 |
| 15.0 | 31 Aug 2024 |

| | | |
|---|---|---|
| bmc | BMC Remedy | 9.1.00, 9.1.02, 9.1.03, 9.1.04, 18.08, 18.05, 19.02, 19.08, 19.11, 20.02 |
| ca technologies | CA PPM | 15.5, 15.5.1, 15.6, 15.7, 15.7.1, 15.8, 15.8.1, 15.9, 15.9.1<br><br>Current On Demand (Cloud) Version |
| | CA Agile Central (Rally) | 2018.1, 2.0<br><br>Current On Demand (Cloud) Version |
| cherwell | Cherwell Service Management<br><br>(Only available for Planview OEM) | 10.0.2 |
| INTLAND SOFTWARE | codebeamer | 8.2, 9.0, 9.1, 9.2, 9.3, 9.4, 9.5, 10.0, 10.1, 20.11 |
| git | Git | All<br><br>*Note: If using a supported |

| | |
|---|---|
| 15.0.46 | 31 Aug 2024 |
| 15.5 | 30 Sep 2025 |

| Release | End-of-Life Date |
|---|---|
| 12.7 | 30 Jun 2022 |
| 12.8 | 31 Aug 2023 |

| Release | End-of-Life Date |
|---|---|
| 12.60 | 31 Aug 2023 |
| 12.60.47 | 31 Aug 2023 |
| 12.60.60 | 31 Aug 2023 |
| 15.0.20 | 31 Aug 2024 |
| 15.0.40 | 31 Aug 2024 |
| 15.0.46 | 31 Aug 2024 |
| 15.0.60 | 31 Aug 2024 |
| 15.1.20 | 31 Aug 2025 |
| 15.1.40 | 31 Aug 2025 |
| 15.1.60 | 31 Aug 2025 |

| Release | End-of-Life Date |
|---|---|
| 9.4 | 31 Oct 2022 |
| 9.41 | 31 Oct 2022 |
| 9.42 | 31 Oct 2022 |
| 9.50 | 30 Jun 2023 |
| 9.51 | 30 Jun 2023 |
| 9.52 | 30 Jun 2023 |
| 9.53 | 30 Jun 2023 |
| 9.54 | 30 Jun 2023 |
| | |

| | | |
|---|---|---|
| | | Git Hosting Service, the version of the service used does not impact functionality. It is used to determine commit location.<br><br>⚠️ Git connector is not available on Tasktop Cloud |
| GitHub | GitHub Issues | Enterprise 11.10.343, 2.3 and higher,<br><br>Current On Demand (Cloud) Version |
| GitLab | GitLab Issues | Enterprise and Community Edition: 10.0, 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.8, 11.0, 11.1, 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 11.8, 11.9, 11.10, 11.11, 12.0, 12.1, 12.2, 12.3, 12.4, 12.5, 12.6, 12.7, 12.8, |

| | |
|---|---|
| 9.55 | 30 Jun 2023 |

| Release | End-of-Life Date |
|---|---|
| 11.5 | 31 Dec 2022 |
| 11.7 | 30 Nov 2023 |

| Release | End-of-Life Date |
|---|---|
| 12.3 | 31 Jan 2023 |

| | | |
|---|---|---|
| | | 12.9, 12.10, 13.0, 13.1, 13.2, 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9, 14.0<br><br>Current On Demand (Cloud) Version |
| IBM | IBM Rational ClearQuest | 9.0, 9.0.1 |
| | IBM Engineering Requirements Management DOORS Family | 9.5, 9.5.2, 9.6, 9.6.1, 9.7, 9.7.1<br><br>⚠ IBM DOORS connector is not available on Tasktop Cloud |
| | IBM Engineering Requirements Management DOORS Next | 6.0, 6.0.1, 6.0.2, 6.0.3 iFix 005 and later, 6.0.4, 6.0.5, 6.0.6, 7.0, 7.0.1, 7.0.2 |
| | IBM Engineering Test Management | 6.0, 6.0.1, 6.0.2, 6.0.3, 6.0.4, 6.0.5, 6.0.6, 7.0, 7.0.1, 7.0.2 |
| | IBM Engineering Workflow Management | 6.0, 6.0.1, 6.0.2, 6.0.3, 6.0.4, 6.0.5, 6.0.6, 7.0, 7.0.1, 7.0.2 |
| | | |

| | | |
|---|---|---|
|  | Jama Connect | 8.36, 8.38, 8.39, 8.42, 8.49, 8.56<br><br>Current On Demand (Cloud) Version |
|  | Micro Focus ALM /Quality Center | On Premise and SaaS versions: 12.5, 12.53, 12.55, 12.60<br><br>14.00-SaaS (Patch 0), 14.01 (SaaS), 15.0, 15.0.46, 15.5 |
| | Micro Focus ALM Octane | 12.53 (inclusive only of 12.53.20 and higher), 12.55, 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<br><br>Current On Demand (Cloud) Version |
| | Micro Focus Dimensions RM | 12.6, 12.6.1, 12.6.2, 12.7, 12.8 |

| | | |
|---|---|---|
| | Micro Focus PPM | 9.4, 9.41, 9.42, 9.50, 9.51, 9.52, 9.53, 9.54, 9.55, 9.62, 9.64 |
| | Micro Focus Solutions Business Manager | 11.3, 11.4, 11.4.1, 11.5, 11.7 |
| Microsoft | Microsoft Azure DevOps Server | 2013, 2013.2, 2013.3, 2013.4, 2015, 2015.1, 2015.2, 2015.3, 2017, 2017.1, 2017.2, 2017.3, 2017.3.1 2018, 2018.1, 2018.2, 2018.3, 2019 RC1, 2019, 2020<br><br>Current On Demand (Cloud) Version |
| | Microsoft Azure DevOps Services | Current On Demand (Cloud) Version*<br><br>*Please note limitations in |

| | | |
|---|---|---|
| | | Connector Documentation |
| | Microsoft Project Server | 2013 SP1, 2016*, 2019*, Project Online*<br><br>*Please note limitations in Connector Documentation |
| | 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 |
| | Pivotal Tracker | Current On Demand (Cloud) Version |
| planview | Planview Enterprise One | 16 On Demand, 17 On Demand, 18 On Demand |
| | Planview LeanKit | Current On Demand (Cloud) Version |
| | Planview PPM Pro | Current On Demand (Cloud) Version |
| | Polarion ALM | 19, 19.1, 19.2, 19.3 |
| ptc | PTC Windchill | 11.1, 12.0, 12.0.1 |
| | PTC Windchill R V&S | 11.0, 11.1, 11.2, 12.0, 12.1, 12.2, 12.3 |
| | QASymphony qTest Manager | 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<br><br>Current On Demand (Cloud) Version |
| | Salesforce: Sales | Current On Demand |

| | | |
|---|---|---|
| | Cloud, Service Cloud, Marketing Cloud | (Cloud) Version |
| servicenow | ServiceNow:<br><br>IT Service Management,<br><br>IT Business Management (Agile Development/SDLC, PPM) | Madrid On Demand, New York On Demand, Orlando On Demand, Paris On Demand, Quebec On Demand |
| | ServiceNow Express | Current On Demand (Cloud) Version |
| SMARTBEAR | 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, 12.11, 12.12, 12.13, 12.14, 12.20, 12.21, 12.31<br><br>Current On Demand (Cloud) Version |
| | Sparx Systems Pro Cloud | 2.0, 2.1.18, 2.1.19, 2.1.20, |

| | | |
|---|---|---|
| | Server | 2.1.21, 3.0, 4.0<br><br>*Premium Editions only |
| | Targetprocess | Current On Demand (Cloud) Version |
| | Tricentis Tosca | 11.0, 11.1, 11.2, 11.3, 12.0, 12.1, 12.2, 12.3, 13.0, 13.1, 13.2, 13.3, 13.4, 14.0, 14.1, 14.2 |
| | Trello | Current On Demand (Cloud) Version - Business Class Edition |
| | VersionOne | Enterprise and Ultimate: 19.2 (Summer 2019), 19.3 (Fall 2019), 20.0 (Winter 2020), 20.1 (Spring 2020), 20.2 (Summer 2020), 20.3 (Fall 2020), 21.0, 21.1<br><br>Current On Demand (Cloud) Version |
| | | |

| | | |
|---|---|---|
| | Whitehat Sentinel | Current On Demand (Cloud) Version |
| | XebiaLabs XL Release | 8.0, 8.2, 8.5 |
| | Zendesk | Current On Demand (Cloud) Version |
| | Zephyr for Jira | 3.2.1, 3.2.2, 3.3, 3.3.2, 3.4, 3.5, 3.6, 4.0, 5.x<br><br>Current On Demand (Cloud) Version |