Turbine User Guide

1. Quickstart

1.1. Quickstart Overview

Welcome to the Turbine User Guide Quickstart section! This section helps you get started with Turbine quickly and efficiently.

What's in Quickstart?

The Quickstart section provides everything you need to begin using Turbine:

Getting Started

- **Getting Started Basics** - Supported browsers, login methods, navigation, and key terms
- **Set Up Your Profile** - Configure your user profile and preferences
- **Best Way to Start**- Recommended learning path and best practices

Turbine Cloud

- **Turbine Cloud** - Cloud-specific features, security, and tenant management
- **Security-Specific Features** - Database security and account security guidance

Try a Solution

- **AI SOC Solution** - A complete, ready-to-use security operations workflow that demonstrates Turbine's capabilities

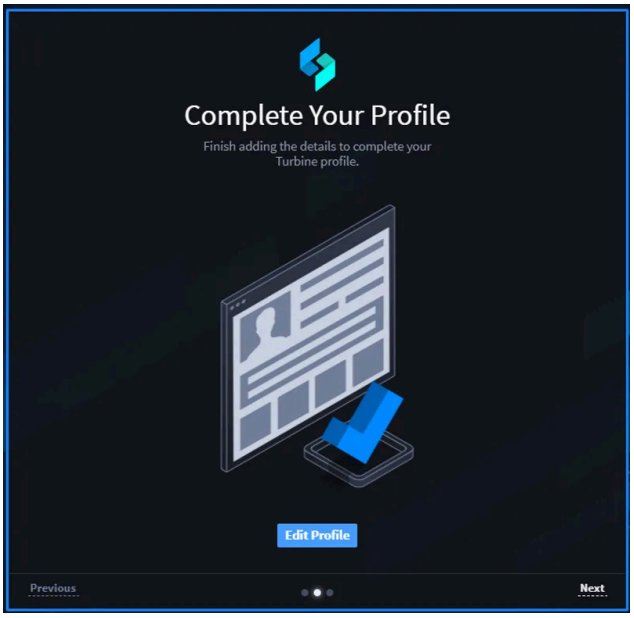
What's Next?

1.1.4.1. Customize Your User Profile

Swimlane Turbine provides flexibility in managing your user profile, allowing you to customize personal settings and manage access tokens, roles, and groups.

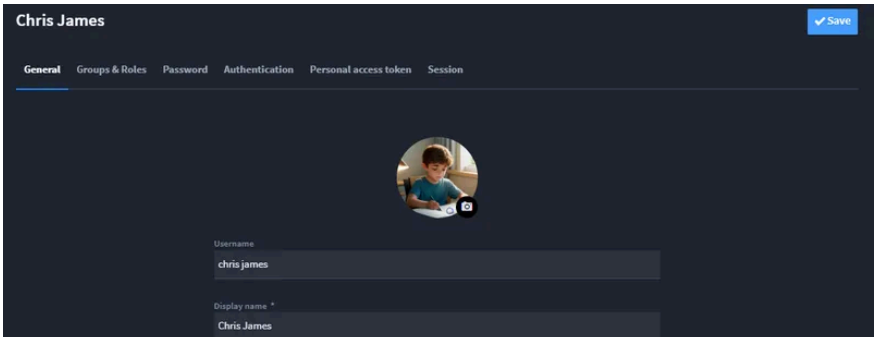
Completing Your Profile

Upon your first login, you'll see the **Complete Your Profile** screen. Follow these steps to finalize your profile:



This action opens the **User Profile Editor**, allowing you to:

- Upload a profile picture to personalize your account.
- View the account's most recent activity.
- Update general details such as your display name, email, and time zone.
- Assign groups & roles to control permissions (Admin only).
- Enable or disable user accounts (Admin only).



2.1.1. Daily Operations Overview

Daily Operations covers all the tasks you perform regularly in Turbine to view, manage, and work with your data.

What's in Daily Operations?

Workspaces

Organize your workspace and navigate between different views and dashboards.

Key Topics:

- Navigate Workspaces and Dashboards
- Workspace management

Dashboards

Create and manage dashboards with charts, visualizations, and interactive elements.

Key Topics:

- Creating and managing dashboards
- Dashboard features (charts, colors, sorting, date ranges)
- Dashboard permissions and sharing
- Dashboard filtering options

Application Records

Work with records in your applications - search, filter, edit, and manage data.

Key Topics:

- **Working with Records** - Search, filter, color code, lock, restrict records
- **Bulk Operations** - Bulk modify, bulk restrict and lock

2.1.1.3.5. Deleting or Editing a User Dashboard

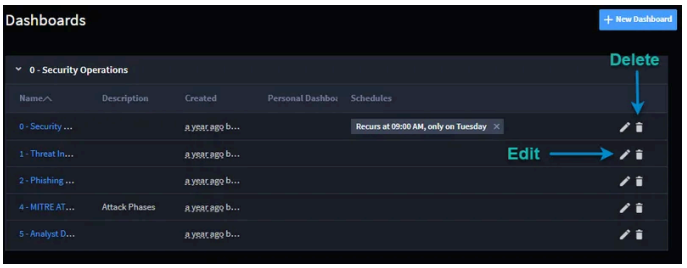
To delete a dashboard:

From the Dashboards taskbar menu, select **Delete**.

You can change the settings of the dashboard by clicking on **Settings and Schedules**. The Dashboard Settings and Schedules pages allows you to edit the following:

- **General Settings**: Edit the **Name**, **Description**, and **Workspaces**.
- **Advanced Settings:** Change the timeline filter duration.
- **Timeline Filters:** Select date fields to apply global date range filters across applications.
- **Schedules:** Create or edit the schedules.
- **Permissions:** Edit the permissions.

You can also edit or delete a dashboard from the Dashboards page. Click the pencil (edit) or the trash can (delete) icon.



2.1.1.3.6. Set Dashboard Permissions

To modify the dashboard permissions, from the Create or Edit Dashboard dialog, click the PERMISSIONS tab. Dashboards can be accessible personally or through role-based access control.

You can also set up private dashboards. Private dashboards allow end-users to create personal dashboards that only they can view and manage.

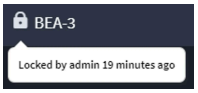
If you have permission, you can also set up private dashboards. A private dashboard can only be viewed or modified by whoever created the dashboard. Administrators or the creator of

2.1.1.4.2.1. Record Lock

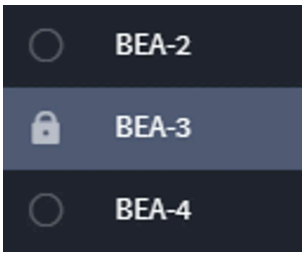
You can lock records while editing and modifying them in order to prevent your changes being overwritten by another user. In addition, locked records can not be bulk modified or bulk deleted.

Once a record is locked, the lock icon appears next to the record's Tracking ID in the Record header.

When a record is locked, only the user who created the lock or an administrator can unlock it. To unlock a record, from the Record header, access the record menu and select **Unlock Record.**

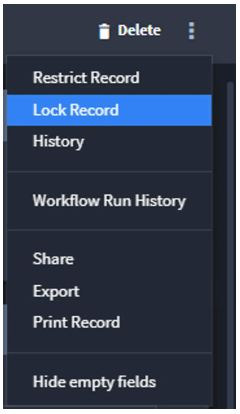


Locked records are also indicated in the report-view-of-records (Default Report) interface.



To lock a record:

1. Open a record. From the record taskbar, access the record menu.



2. From the menu, select **Lock Record**.
   The record is locked immediately. Anyone can access the individual record and open it, but only the user who created the lock or an administrator can make changes to the

2.1.1.4.5. Advanced

2.1.1.4.5.1. Lookup and Create References within Records

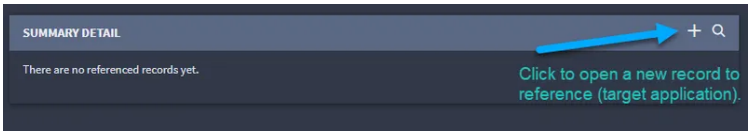
You can select which field(s) to reference from within a record. A reference field on a record, whether single-select, multi-select, or grid, is accompanied by a New and Lookup button adjacent to the reference field.

For more information about reference fields in general, see **Reference**.

Creating a New Record to Reference

To create a new record to reference:

1. From within an open record, locate the reference field, and then click **+**, or Add New. A record editor for the target application opens.



2. Fill out the target record data.

Looking Up References within Records

To lookup and create references within records:

1. Click within the field, or click the Lookup (Search) icon. Enter a keyword or search term and then press Enter to begin the search.

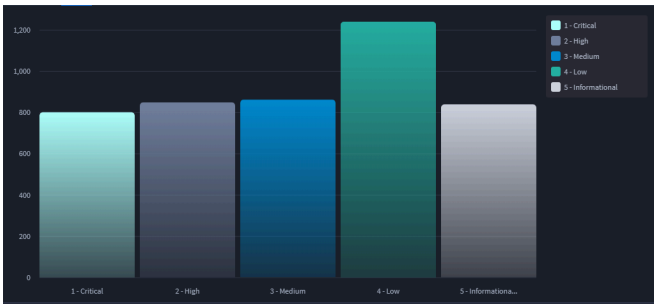
2.1.1.5.4.2. Chart Types

This topic contains chart type examples.

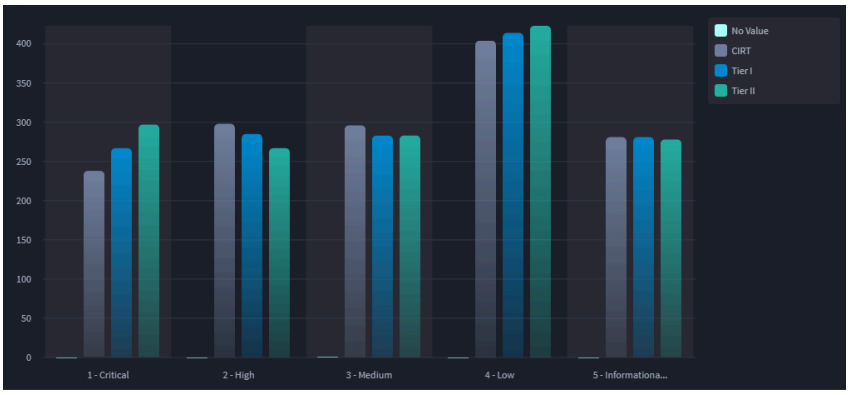
Bar Charts

Use bar charts to show comparisons between different categories of data. The bars can display either vertically or horizontally.

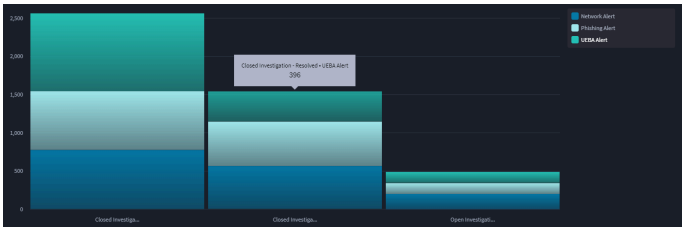
Vertical Bar



Grouped Vertical Bar



Stacked Vertical Bar



100% Vertical Bar

2.1.2.1.1. Getting Started

2.1.2.1.1.1. Playbooks

Playbooks are a series of triggers, logic, and actions that automate a workflow. A playbook can contain triggers, actions, native actions, components, assets, inputs, and outputs.

Playbook Architecture

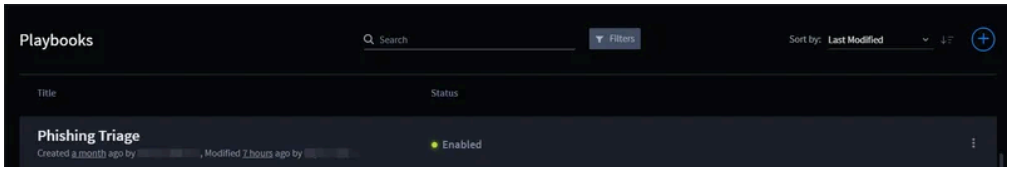
- A playbook can have one or more flows.
- Each flow has exactly one trigger.
- Flows do not communicate with each other.

For details, see **Flows**.

Playbooks Home Page

From **ORCHESTRATION > Playbooks**, you can:

- Search for playbooks by keyword
- Filter by source, interface, or creator
- Sort by last modified, last created, or alphabetical order
- Enable/disable playbooks
- Open playbook operations (export, duplicate, delete)



2.1.2.1.2.1.5. Using Webhook Triggers in Swimlane Turbine

Webhooks in Turbine enable seamless real-time communication by facilitating data exchange between third-party services and Turbine records. Webhooks act similarly to API callbacks, allowing playbooks to receive and process data without complex polling mechanisms. As an orchestrator, you can easily configure a webhook, generating a unique URL to integrate a playbook with web applications for automated data processing.

Viewing Existing Webhooks

To view existing webhooks in Turbine:

1. Navigate to **ORCHESTRATION** in the platform.
2. Click **Webhooks**.

You can toggle the **Disabled** button to enable or disable a webhook. Click a webhook to view or modify its settings and logs. The **Filter by name** option helps you quickly find a specific webhook.

Create Webhook Triggers

One common use of webhooks is to ingest or push data, such as importing alerts into a playbook. To configure a webhook trigger:

In a playbook, from the Add panel, click and drag **Catch Webhook** to the canvas.

Hover over the plus icon to add it to the canvas. The Trigger panel displays to the right of the canvas, where you can configure the webhook trigger.

Edit Webhook Settings

To configure webhook-specific settings such as the URL or authentication:

1. In the Trigger panel, click Edit Webhook.
2. In the **Settings** tab:
    - Copy the **Webhook URL** to use with external systems.
    - Add or update the description if needed.

2.1.2.1.3.5.1. Create Record

The **Create Record** native action creates a new record in a selected application from the playbook builder. The action maps data to application fields and returns the tracking ID of the created record.

Overview

The Create Record action enables you to programmatically create records in Swimlane applications from within playbooks. This is useful for automating record creation based on events, data from other systems, or calculated values from upstream actions.

Key Benefits

- **Native Action:** Integrated directly into the playbook builder for streamlined workflow creation.
- **Flexible Field Mapping:** Map data from upstream actions, playbook inputs, or expressions to application fields.
- **Record Restrictions:** Optionally restrict record access to specific users or groups.
- **Tracking ID Output:** Returns the tracking ID of the created record for use in downstream actions.
- **At Most Once Execution:** Ensures the record is created exactly once, even if the playbook is retried.

Inputs

The Create Record action requires the following inputs:

Required Inputs

- **Application:** Select an application by either:
  - **Application ID**: The unique identifier of the application
  - **Application Name**: The name of the application
- **Note:** You must provide either `applicationId` or `applicationName`, but not necessarily both.

2.1.2.1.3.11. Using the Parallel Native Action

The **Parallel** native action in Swimlane Turbine allows orchestrators to execute multiple paths in parallel, waiting for all branches to complete before continuing downstream. This action is ideal for workflows requiring simultaneous processing of tasks with synchronized results.

Overview

The Parallel native action enables simultaneous execution of multiple actions, optimizing workflows that require concurrent processing. By ensuring all parallel tasks complete before proceeding, the action provides precise control over complex playbook flows.

Key Benefits of the Parallel Action

- **Native Action:** Integrated directly into the playbook builder for streamlined workflow creation.
- **Concurrent Processing:** Enables simultaneous execution of multiple actions, reducing overall processing time.
- **Synchronized Results:** Ensures that all branches complete before the workflow continues, maintaining workflow consistency.
- **No Connector Needed:** Operates within Swimlane without external integration.
- **WAIT Feature:** Provides a visual indicator showing that the playbook is waiting for parallel branches to complete before proceeding.

Understanding the Parallel Action

Using the Parallel native action, you can execute multiple paths simultaneously, wait for them to complete, and then continue the playbook flow. The action provides **On Success**, **On Failure**, and **On Complete** paths based on the outcome of the parallel branches.

How Parallel Actions Work

1. **Entrypoints:** All entrypoint actions defined in the Parallel group are executed simultaneously when the Parallel action starts.
2. **Conditional Entrypoints:** Entrypoint actions can have conditional `if` expressions. If

2.1.2.1.6.1. Inputs

Actions are individual capabilities of connectors that interact with external technologies by passing in data via action inputs. Results are available from action outputs.

Actions can:

- Call another playbook
- Interact with external technologies
- Contain inputs and outputs (outputs can be promoted through the **Playbook Outputs** dialog)

> You do **not** need a trigger to add an action to your playbook.

Create Actions

1. From your current playbook, open the **Add** panel on the left side of the screen. The **Add** panel displays available actions, connectors, and components organized by category. The panel has three tabs: **Triggers**, **Actions**, and **Components**.
2. Browse or search for the desired action from the available connectors and native actions.
    - Native actions (such as Create Variables, Update Variables, HTTP Request, Script, Transform Data, Condition, Loop, Parallel, Create Record, Delete Record, Search Records) appear at the bottom of the **Add** panel
    - Connector actions are organized by vendor/connector above the native actions
3. Drag the action from the **Add** panel.
4. Drop the action onto the canvas:
    - Drop onto a drop zone (indicated by a plus icon) to add it to the flow
    - Drop onto an edge to insert it between existing actions
    - Drop onto a node to connect it after that action
5. After dropping, the action appears on the canvas and the **Action** details panel automatically displays to the right of the playbook.
6. In the **Action** details panel, under the **Info** section:

2.1.2.2.3. Assets

Term	Definition	Characteristics
Assets	Saved credentials and key/value pairs that help you connect to technologies, and also serve as a key store for commonly used sets of keys and values	• Can be for a specific connector or customized, which can be applied to any action inputs. • Most useful for standardizing and securing configurations.

Assets can be for a specific connector or customized, which can be applied to any action inputs, and are most useful for standardizing and securing configurations. In Turbine Canvas, assets can be configured in your playbook allowing you to configure a connector and its asset without leaving the playbook canvas.

When to Use Assets

- You need to reuse credentials or common settings across playbooks.
- You want to avoid storing secrets directly in playbook inputs.
- You want consistent configuration for connector actions.

Security Tips

- Use least-privileged credentials for assets.
- Rotate secrets on a regular schedule.
- Review which playbooks depend on an asset before changing it.

Key Benefits of Assets

- **Standardization**: Assets ensure that sensitive data, such as API keys, credentials, and other key-value pairs, are consistently applied across different playbooks, avoiding manual input errors.
- **Security**: By centralizing credentials and keys in assets, they can be managed and

2.1.2.4.1. Classic Playbooks

Playbooks are where automation is built quickly and easily and enriches data processing. With Swimlane Turbine's playbooks, anyone can create modular, repeatable automations that process real-time data.

What are Playbooks?

Playbooks are self-contained modular entities that help you quickly and easily build and/or enrich automation processes.

Playbooks contain:

- Triggers
- Actions
- Conditions
- Inputs and Outputs
- Repeats
- forEach Loops

Playbooks are initiated by triggers and used to achieve desired outcomes.

Playbook Inputs

Playbook inputs pass and normalize data into a playbook from a trigger, application, or event.

Triggers

One or more triggers automate actions based on the criteria set within a playbook. Swimlane Turbine currently uses four types of triggers to retrieve and/or ingest data.

1. Webhooks
2. Record events
3. Schedules
4. Record action button

Once a playbook has been triggered, a series of actions within that playbook are executed

2.1.2.4.1.5. Nested Playbooks

You can nest playbooks into other playbooks as actions. Refer to **Actions** for instructions to add an action to a playbook.

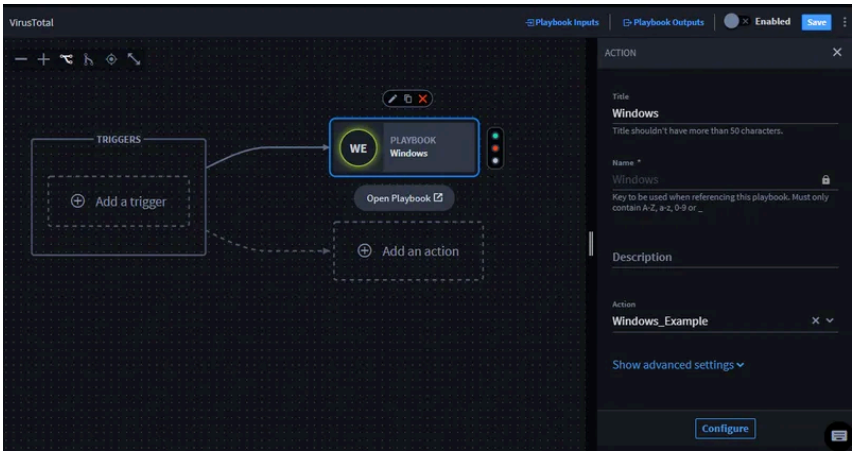
Terms

Term	Meaning
Parent playbook	A playbook that calls another playbook.
Nested playbook	A playbook that calls another playbook as an action.

> When nesting a playbook into a parent playbook, the nested playbook's inputs and outputs are available to map to and use in the parent playbook.

To nest a playbook to a new/current playbook:

1. Navigate to the **Playbooks** home page, and then open a playbook (or create/upload a playbook).
2. Click **Add an action**.
3. From **ACTION**, click the **Action** drop-down menu, and select an existing playbook.
4. From **ACTION**, add a title.



5. Click **Configure** to map inputs and outputs.
6. If the nested playbook has inputs and/or outputs, they are available on **Action Inputs**.
7. Review **Action Outputs**.

2.1.2.4.1.8.7. Classic Retries

Playbooks allow you to set action conditions that automate retrying actions based on conditions, time limits, and retry count attempts. Retries allow you to set up the following criteria for an action:

- Interval
- Frequency
- Number of attempts/tries

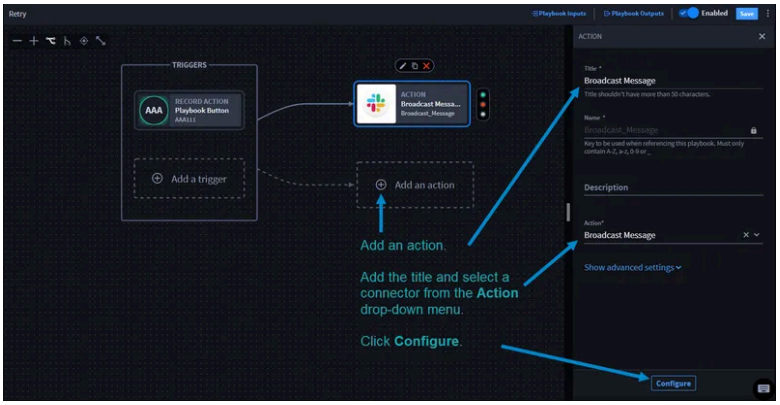
After you set the criteria, you can manage the parameters of that retry by adding conditions. The retry condition settings are similar to how you **Create Action Flows and Conditions**.

> Conditions are **not** required.

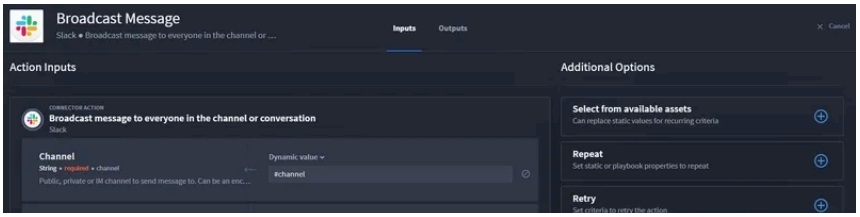
Apply Action Retries

To retry an action:

1. Add and configure an action.



2. From ACTION, click **Configure**.
   Here is where you will configure the action inputs, and then set up the action retry by entering appropriate data in the input fields to set your parameters.



2.1.2.4.1.9.5. Scripts

As analysts and security professionals, you have knowledge on writing simple scripts in Python. And while Turbine allows you to write more complex code with JSONata, you can also use a native action for simple functions.

> For Boolean and Null data types, we recommend importing JSON and using **json.loads(<my ref>)** to ensure the data loads correctly. Since all playbook data is JSON and Python does not support all JSON types natively.

Use the controlled Script native action and write with Python to:

- manipulate data and edge cases.
- reduce complexity used with JSONata.
- use the most common programming language in security today to do simple tasks.

Turbine uses Python 3.11 and accesses all the standard libraries that come with Python, including the **Python Standard Library**. You can also use **Numpy** version 1.25.2, and **Pendulum** version 2.1.2.

When configuring inputs, consider using the Swimlane Python Chatbot, which uses ChatGPT's Open AI to help you formulate transformations and customized Python code. See **Swimlane Python Chatbot** for information.

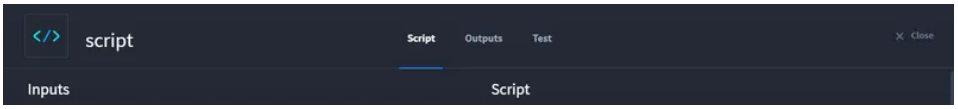
Script Native Action Set Up

Time to start basic set-up for the Script native action.

You have already created a playbook, and you are ready to manipulate data from a property.

1. From your playbook, click **Add an Action**.
2. From the ACTION panel, click the **Action** drop-down menu.
3. Select **Script**, then click **Configure**.

The Script window opens.



2.1.2.4.1.10.5. Webhook Triggers

Webhooks are a type of event stream that enables products, vendors, or services to push real-time communication in Turbine. They process and enrich data to send from third-party services and platforms to Turbine records.

> Click **Save** or **Save and Close** after making changes.

View Existing Webhooks

To view existing webhooks:

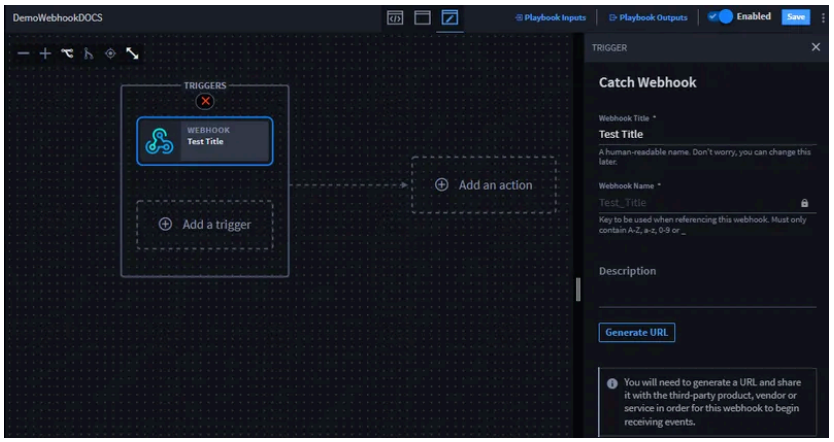
1. Navigate to **ORCHESTRATION**.
2. Click **Webhooks**.

From here you can toggle the **Enabled** button to enable or disable the webhook, or click the desired webhook to view settings and log information.

Create Webhook Triggers

To create a webhook trigger, create a new playbook, upload a playbook, or search for an existing playbook.

1. On the playbook, click **Add a trigger**.
2. Click **Catch Webhook**.
3. On TRIGGER, enter a webhook title and click **Generate URL** or the webhook will not be created.



2.1.3.1. Getting Started

2.1.3.1.1. Applications and Applets

Applications and Applets are the foundation of the Swimlane Turbine platform.

What is an application?

A user-defined template for collecting, storing, and organizing your data. All automated activities and decisions are driven by how your application stores data. You also manage workflow from within applications.

What is an applet?

A preconfigured set of fields and layout specifications. Applets are appended to an existing application form layout and allow users to easily update and expand their existing applications.

This section lists all the Applications and Applets that have been created and are available on this Turbine instance. You can filter the list by type (Application and Applets), or by a search string that you enter.

Each Application and Applet provides a management menu with viewing options, copy and export options, and a delete option.

To review or reconfigure the application, click **Builder**.

2.1.3.1.2. Manage Applications and Applets

The Applications and Applets home page is where you manage your applications and

2.1.3.2.6.1. Attachments

Use this field to store files in a record. The list of preview options are: .csv, .doc, .docx, .gif, .jpeg, .jpg, .pdf, .png, .ppt, .pptx, .tsv, .txt, .xls, and .xlsx. There are no document-type limitations for attachments.

To create an attachments field:

From Application Builder's Field Types, select **Attachments** and then drag and drop it to the Form Layout. Drop the field in the layout area, or within a tab or section layout object.

Access the field's properties and complete the following fields as needed:

Field	Step	Example
Display Name	Enter the name of the field.	*Related Files*
Help Text	Enter contextual help text. You will first need to specify whether the help text will appear above or below the field in the record form, and then you can enter the text.	*Attach related files here.*
Read-only	Click to indicate that the field is read-only for the record. The field will not be editable.	*checkmark*

Next, consider the following advanced options in the FIELD PROPERTIES tab:

Field	Step	Example
Max Size	Use to specify the maximum allowed file size for the attachment.	*100,000 kb*
Delete Attachments	Select this field to indicate that the attachment be deleted.	*select*
Time to live (days)	Use to indicate the number of days before the attachment is deleted.	*30 (days)*

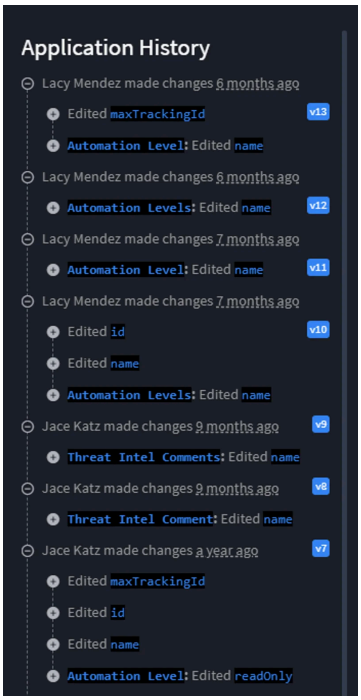
Add specific field-level permissions by role, if needed, and then click **Apply.**

2.1.3.2.7. View Revision History - Application

Every modification to the application's layout is stored in the revision history.

- Adding/Removing fields
- Adding/Removing layout elements
- Field configuration changes

To view revision history, access the pull-down menu from the Application Builder taskbar and select **History**.

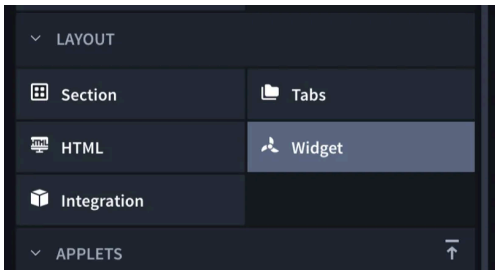


Revisions are listed by version number and show which user made the change and which fields were affected. Expand the revision to see additional detail.

2.1.3.3. Building Applets

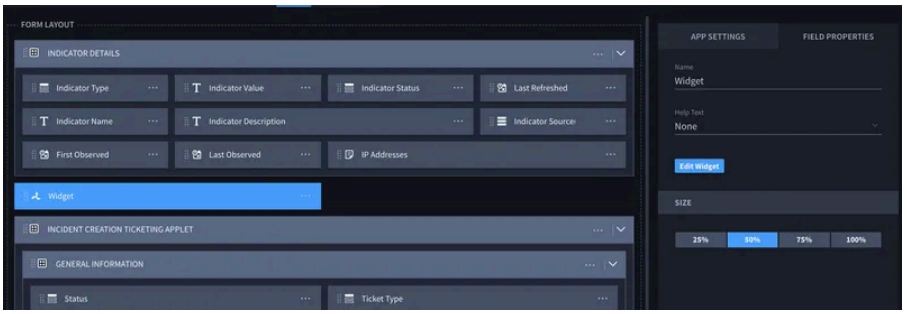
2.1.3.4.3. Create Widgets

To create Widgets, from the **Layout** section of the **Application Builder or Applet Builder** page, select **Widget** and then drag and drop it into the Form Layout.



> You can add multiple widgets per application or applet.

To edit the widget, select the widget in the layout. Then, in the properties sidebar, click **Edit Widget.**



Widgets are rendered on the record page and have access to the record values through the record attribute. Every change to the record will automatically update the widget, by calling its `update` method.

You can trigger playbooks from widgets. The application from which you are building the widget must contain a playbook button. Use the button property in context data to trigger a playbook upon clicking the playbook button. This reuses the playbook inputs specified in the playbook.

Use this code to set up a playbook button widget:

```
import { SwimlaneElement, html } from '@swimlane/swimlane-element@2';export
default class extends SwimlaneElement {
  handleClick() {
    this.triggerButton(this.contextData.application.buttons[0].id);
```

2.1.3.5.6. Workflow Best Practices

Best Practices

Workflow Design

1. **Start Simple** - Begin with basic conditions and actions, then add complexity
2. **Use Descriptive Names** - Name stages and actions clearly
3. **Document Complex Logic** - Add comments or documentation for complex workflows
4. **Test Incrementally** - Test each stage as you build it

Performance Considerations

1. **Limit Condition Complexity** - Avoid overly complex nested conditions
2. **Optimize Field References** - Reference fields efficiently
3. **Use Appropriate Operators** - Choose operators that match your data types
4. **Avoid Circular Dependencies** - Don't create workflows that reference themselves

Maintenance

1. **Review Regularly** - Periodically review workflows for optimization
2. **Document Changes** - Keep track of workflow modifications
3. **Test After Changes** - Always test workflows after modifications
4. **Version Control** - Use application versioning to track workflow changes

Common Patterns

Pattern 1: Progressive Disclosure

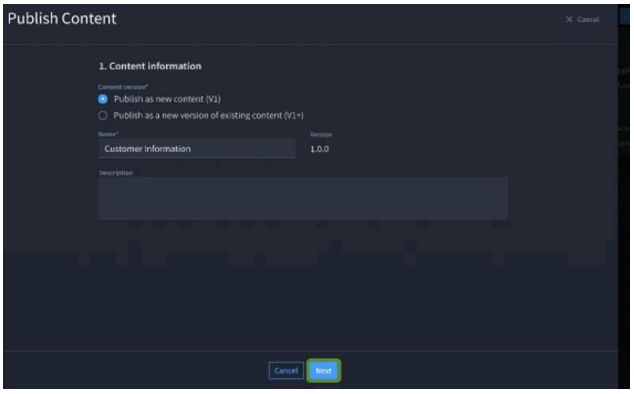
```
If Record Type = "Incident"
  ├── Show Incident Fields
  └── If Severity = "High"
      └── Show High Priority Fields
```

2.1.3.7. Publishing

2.1.3.7.1. Publishing an Applet

An Applet can be published to the Turbine Library as content that can be shared across your Turbine account. The publishing process includes:

1. **Publish as New Content (V1)**
2. **Publish as a New Version of Existing Content (V1+)** (including handling merge conflicts if a remote repository has been configured)



Publish as New Content (V1)

This process applies when the applet is being published for the first time.

Steps:

1. Open the applet settings menu and select **Publish Applet** .
2. In the **Publish Content** window:
3. Select **Publish as New Content (V1)**.
4. Provide a name and description for the content.

2.1.5.3.1. Groups

In Swimlane Turbine, Groups serve as collections of users with shared permissions, roles, and access levels, allowing administrators to manage access and permissions efficiently across multiple users. Each user can belong to several groups simultaneously, which is particularly useful for organizing users by department, role, or specific project needs. By managing permissions at the group level, organizations enhance security, streamline onboarding and offboarding, and ensure consistent access control across departments, projects, and workflows, ultimately saving time and improving collaboration.

How Groups Work

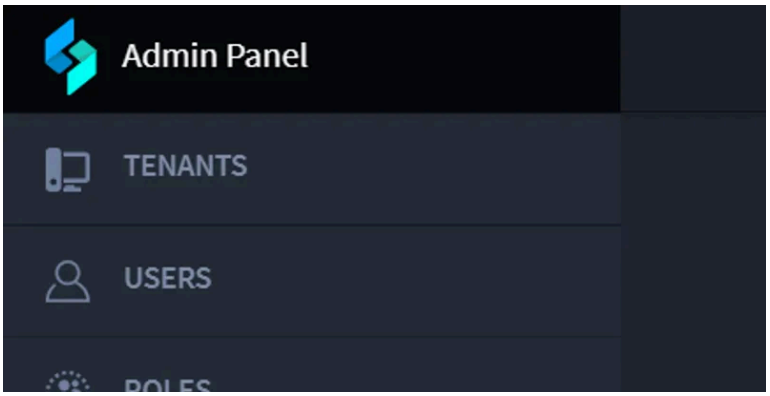
Each user can belong to single or multiple groups simultaneously, making it easy to organize users based on department, role, or specific project needs. This flexibility supports dynamic roles and cross-functional team structures.

Key Benefits:

- **Enhanced Security:** Permissions are managed at the group level, ensuring users have access only to relevant resources and workflows.
- **Streamlined Onboarding and Offboarding:** Adding or removing users from groups automatically applies the appropriate permissions, simplifying user lifecycle management.
- **Consistency and Efficiency:** Group-based management ensures consistent access control across departments, projects, and workflows while reducing administrative overhead.

Access the Groups Page

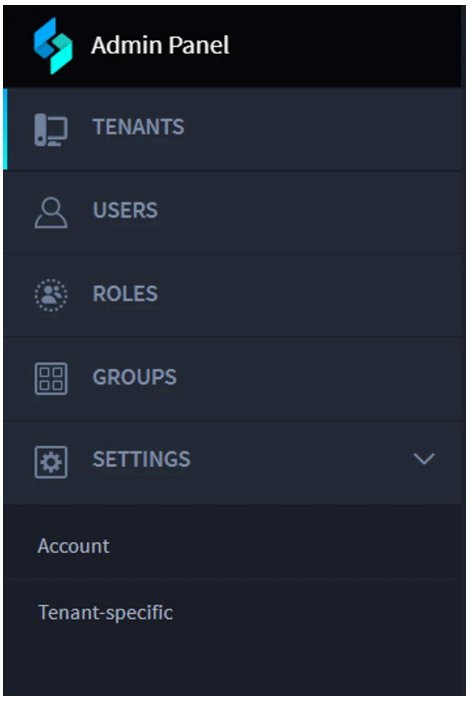
To access the **Groups** page, navigate to the Admin Panel and select **Groups**.



2.1.5.4. Settings

Swimlane Turbine Settings control global Turbine settings and contain error and troubleshooting tools. Turbine Settings are only available to administrators.

Click **SETTINGS** in the Admin panel menu to access the **Account** and **Tenant-specific** settings.



In Settings, administrators can:

- Back up Turbine.
- Set the Turbine URL.
- Update the Turbine license.
- Sync Turbine with Active Directory or Open LDAP.
- Define security parameters.
- Specify the Turbine administrator email.
- Set up an outgoing email server.
- Format outgoing email messages.
- View Turbine logs and the background jobs dashboard.

2.1.5.4.1. Account Settings Overview

2.1.5.5.1. Enable SAML for SSO

SAML (Security Assertion Markup Language) is an open standard that facilitates single sign-on (SSO) by allowing Swimlane Turbine (the service provider) to rely on an external identity provider (IdP) to authenticate users.

Using SAML, there are two possible ways to initiate login:

- **Service Provider-initiated:** The login process starts with Swimlane Turbine.
- **Identity Provider-initiated:** The login process starts with the external identity provider (for example, Okta, JumpCloud, and Azure Entra).

Benefits of SAML Authentication

SAML authentication provides numerous advantages for organizations, improving security, streamlining user access, and simplifying administration. Below are the key benefits:

- **Centralized User Management**: User authentication is managed through a single, external IdP, reducing administrative overhead.
- **Enhanced Security:** Leverages the security features of the IdP, such as Multi-Factor Authentication (MFA) and conditional access policies.
- **Seamless Login Experience:** Users can log in to Swimlane Turbine using existing credentials, improving usability.
- **Scalability:** Easily accommodates user management for large organizations or multi-tenant environments.

Turbine Service Provider SAML Metadata

The table below provides key metadata information that you will need when configuring Swimlane Turbine as a service provider with an external identity provider:

Metadata	Description	Usage
Service Provider (SP)	The unique identifier for the service provider (Swimlane Turbine). This value must be entered in the identity provider's configuration in the IDP to recognize Turbine as a	`https://{swimlane-hostname-here}/tenant/api`

2.1.5.7. Monitoring and Dashboards

2.1.5.7.1. Usage Dashboard

The **USAGE** tab within the Admin Panel displays the usage metrics for Playbook runs, Hero AI, and Events across the tenants. You can configure the dashboard and filter the data using the filters provided.

For more information, see the following:

- **Playbook Runs Dashboard**
- **Actions Dashboard**
- **Events Dashboard**
- **Notifications Dashboard**
- **Hero AI Dashboard**

2.1.5.7.1.1. Playbook Runs Dashboard

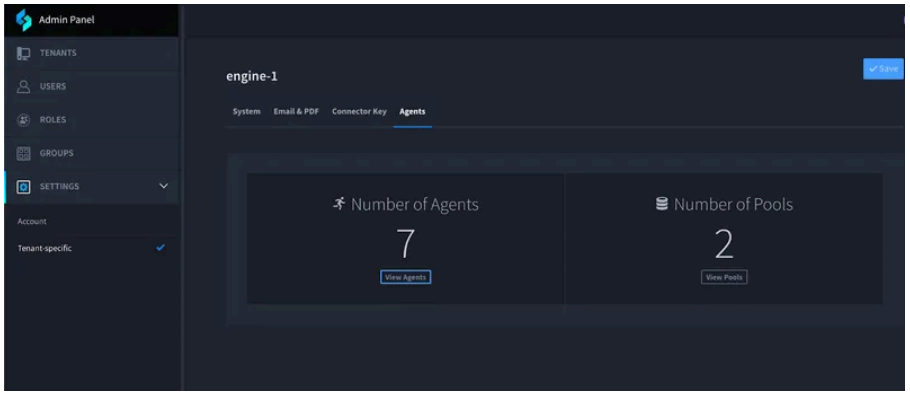
The **Playbook Runs** dashboard helps in tracking the number of successful playbook runs across the tenants. The following metrics are displayed:

- **Total Playbook Runs:** Displays the total number of playbook runs for the selected date range, tenants, playbook types, trigger types, and playbooks.
- **Total Playbook Runtime:** Represents the cumulative runtime of all playbook runs within

2.1.5.8.1.2. Installation

Install the Agent:

1. In Swimlane UI: **SETTINGS > Tenant-Specific > Agents**



a

2. Click **View Agents** to see your agents and pools.
3. Click **Deploy Agent** to open the install window.
   Note: Do not click outside this window or it will close and you must start again.
4. Fill in required fields (see below for field details).
5. Click **Copy install script**.
6. Paste the script into your terminal and run it.
7. Answer any prompts or questions to finish installation

Sample Install Script

```
curl -k https://<your-turbine-instance>/agent/install.sh | bash -s --
AGENT_NAME=agent-xyz
```

The script prompts for your Turbine username and password. Credentials are not saved.

Configuration Options

Agent Install Script Fields:

Only Required Field	Information

2.1.5.8.5. Uninstall Remote Agent

To uninstall the remote agent:

- Delete the two agent containers in Docker Desktop / Podman:
  - `docker rm -f <agent_name>-turbine-agent / podman rm -f <agent_name>-turbine-agent`
  - `docker rm -f <agent_name>-podman / podman rm -f <agent_name>-podman`
- After the remote agent is disconnected (may take a couple of minutes after container deletion), delete the remote agent from the **Tenant-specific** settings page, list of tenants.
- `<agent_name>` is the name of the agent user input while creating the agent.

2.1.5.9. Audit and Compliance

2.1.5.9.1. Audit Logging

Audit logging in Swimlane Turbine records critical actions and events across the system, providing visibility for monitoring, security, and compliance. Audit logs can be retrieved using the API, and users can apply filters to narrow down the results based on account, tenant, and time range.

Understand the API Log Retrieval Endpoint

Audit logs in Swimlane are accessible via two endpoints:

2.1.6.2.2. Turbine Schema Reference (Classic SOC)

> Turbine Schema was formerly known as Turbine Extensible Data Schema, or TEDS.

This document describes the Turbine Schema objects used by the **Classic SOC Solutions Bundle**, including Alert, Email, Observable, Enrichment, and supporting objects.

For general concepts, best practices, and troubleshooting, see **Working with Turbine Schema**. For the AI SOC Solution schema with additional alert, email, and search-params fields, see **Turbine Schema Reference (AI SOC)**. For Classic SOC interface contracts, see **SOC Interfaces**.

Schema Examples

The following examples show real-world Turbine Schema-formatted data structures used in the SOC Solutions Bundle.

Alert Object Example

Here's a complete Alert object following Turbine Schema conventions:

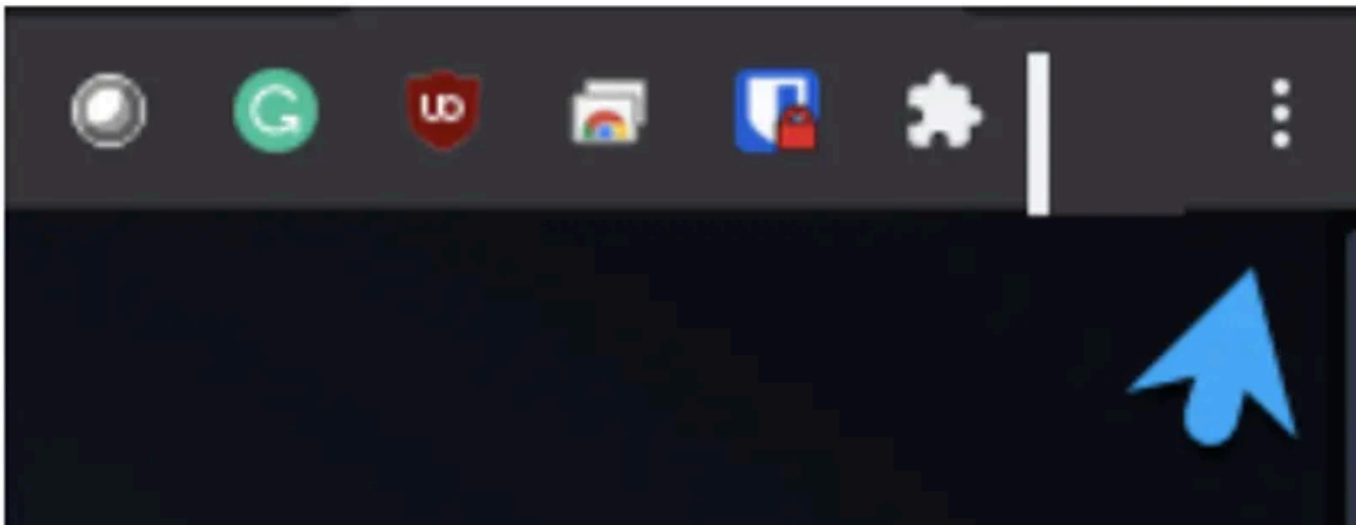

```
{
  "alert_uid": "splunk-alert-20250115-001234",
  "alert_title": "Suspicious PowerShell Execution Detected",
  "alert_description": "PowerShell script executed with encoded commands on host WORKSTA
  "alert_severity": "High",
  "alert_created_timestamp": "2025-01-15T10:30:00Z",
  "alert_start_timestamp": "2025-01-15T10:28:15Z",
  "alert_end_timestamp": "2025-01-15T10:30:00Z",
  "alert_ingested_timestamp": "2025-01-15T10:30:05Z",
  "alert_provider": "Splunk Enterprise",
  "alert_organization": "Acme Corporation",
  "alert_categories": ["Malware", "Execution"],
  "alert_impacted_hostnames": ["WORKSTATION-01"],
```

3.2.5. How to generate a HAR file (Chrome)

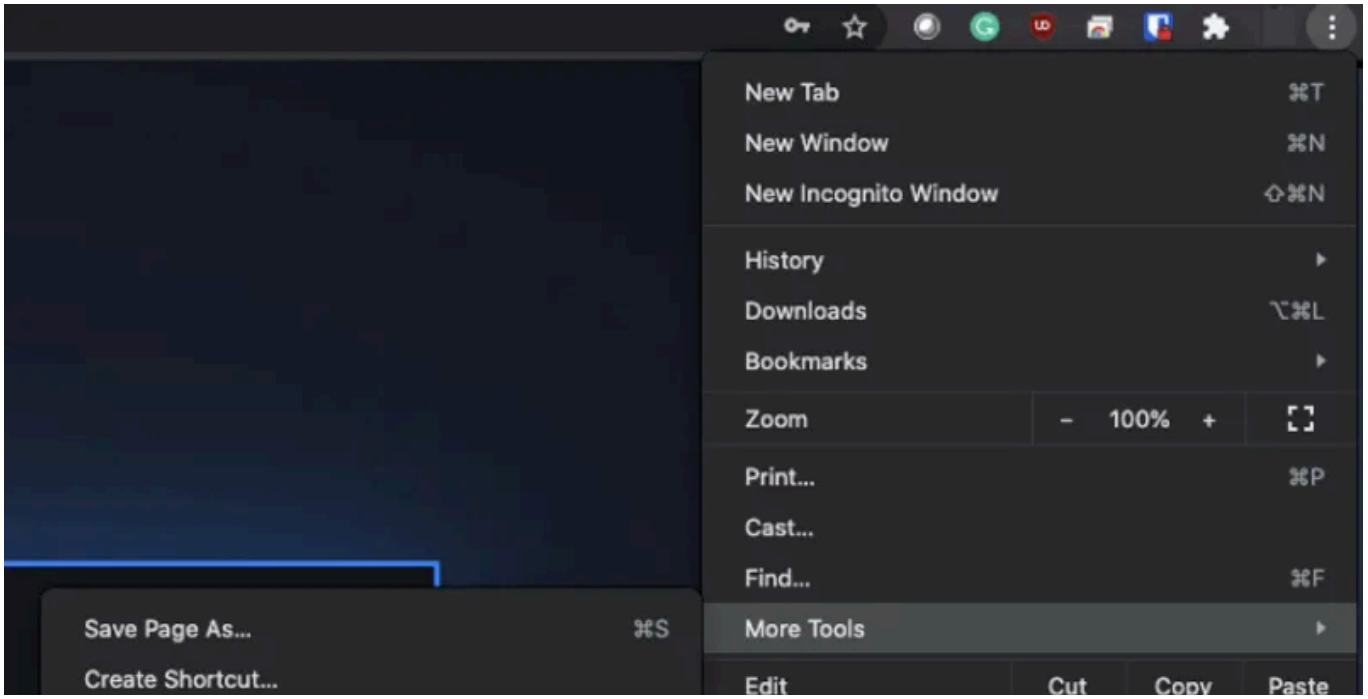
When there are issues with Swimlane, it is sometimes important to acquire the front-end error logs for the platform. To do so, we have to record a HAR file, which is a log of network requests. Be aware that this file will capture all network traffic on that specific browser tab and export it. Only capture web data that you want to be shared.

To capture a HAR file on Chrome:

1. Open Chrome, browse to the page with the issue
2. Go to the vertical ellipsis at the top right



3. More tools > Developer tools



3.3. Use Cases

3.3.1. Condition and Logic Use Cases

3.3.1.1. Configure Inputs in Condition Expressions

Turbine allows users to configure conditions between two actions by utilizing playbook inputs and/or action outputs.

Scenario

Alex wants to configure a conditional expression using playbook property inputs. Alex already has created a playbook and selected an action. She can now configure the action input by adding a condition.

To add a condition, from an action, click the **Add condition** icon.
Once Alex clicks the action flow, it turns blue and the FLOW panel displays to the right.

3.3.2.1. HTTP Requests

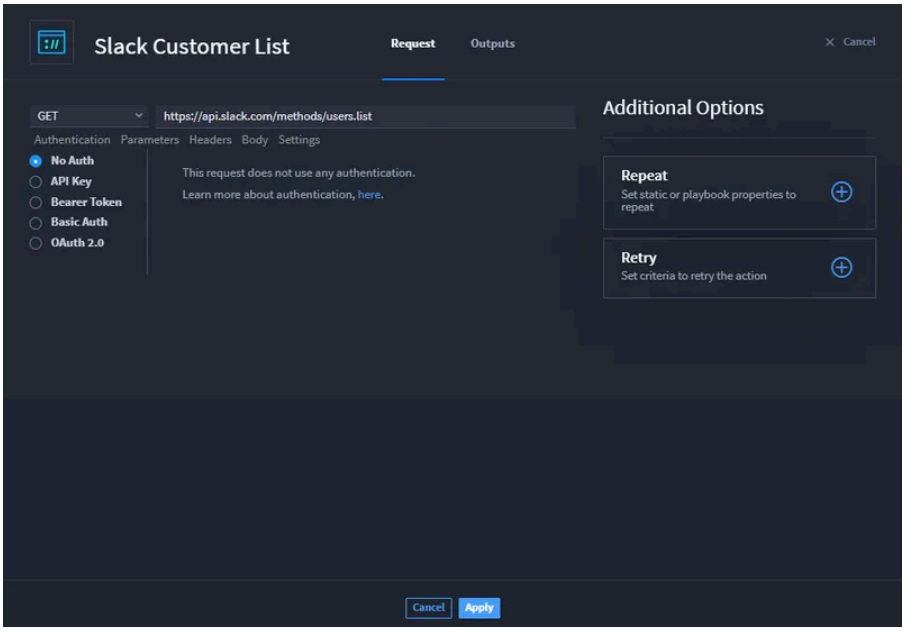
Use HTTP Request native actions to get data from or send data to an API endpoint, even if the connector does not exist for the service. This gives flexibility in integrating Turbine with third-party services.

Scenario: Get Users from Slack

Max, a Turbine Admin, needs to verify a list of Slack users via the Slack API. He uses the **HTTP Request** action to retrieve data from the API using a **GET** request.

Steps to Set Up a GET HTTP Request:

1. Select **No Authorization** under the Authentication tab.
2. Enter the URL: **https://api.slack.com/methods/users.list**.
3. On the Settings tab, ensure that the **Enable SSL certificate verification** toggle is ON.
4. Click **Apply**.



> It's a best practice for **GET** requests to not include a request body.

Conclusion

Max successfully retrieves the Slack user list using the GET method, without needing

3.3.5.1. Script Test Use Case

Scenario: Test Script Action

Rowan is an orchestrator who wants to map static and playbook property data to the Script native action inputs so that he can reference this data in the Python script and test the inputs to see the results before continuing to build the playbook. He also wants to see if there are any discovered outputs in addition to the original outputs.

First, Rowan adds his webhook trigger, so the webhook property data is available downstream in the playbook. Let's see how he sets up the Script native action to see if there are **Discovered Outputs and Testing**.

Steps to Configure the Script Native Action:

1. Click the **Action** drop-down menu, select **Script**, and enter a title for the action.
2. Click **Configure** to open the Script configuration window. This window displays three tabs: **Script**, **Outputs**, and **Test**.
   Note: To ensure proper testing, verify that your script has valid inputs and a valid output structure. This avoids errors in the next steps.
3. From the Script tab, click **Add property** and select **String**. Enter the value "**hello**" for this property.
   Tip: You can add multiple input types (e.g., strings, numbers, objects) to test various scenarios.
4. Add another property of type **Object**. Rename the properties as **string_test** and **object_test1** for easier identification.
   Inside the **Object** property, click **Add property** again and select **String**. Name this inner string property **string_2**.
5. In the **Script** pane, enter the script:
6. Navigate to the **Test** tab. This allows you to test the script before finalizing the action.
7. If there are any **discovered outputs** (outputs automatically generated but not initially visible), click the checkbox next to the discovered outputs to include them for downstream actions.
   Important: Discovered outputs can be added for further use in the playbook. Review discovered outputs thoroughly to avoid missing important properties.