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

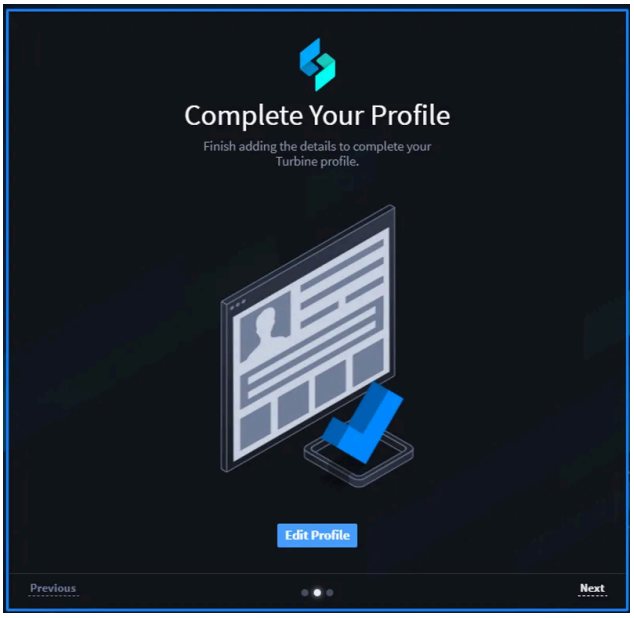
What's Next?

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.

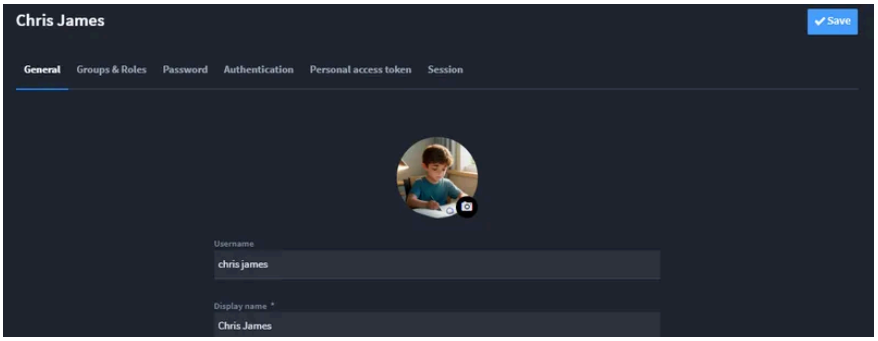
Completing Your Profile

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

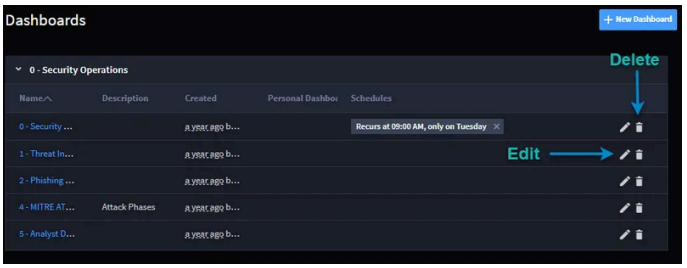
To delete a dashboard:

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.

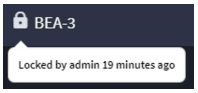
If you have permission, you can also set up private dashboards. A private dashboard can only be viewed or modified by whoever created the dashboard. Administrators or the creator of

2.1.1.4.2.1. Record Lock

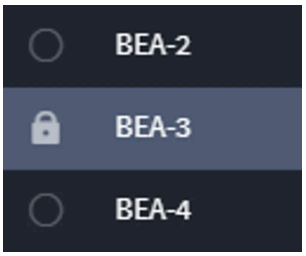
You can lock records while editing and modifying them in order to prevent your changes being overwritten by another user. In addition, locked records can not be bulk modified or bulk deleted.

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.**

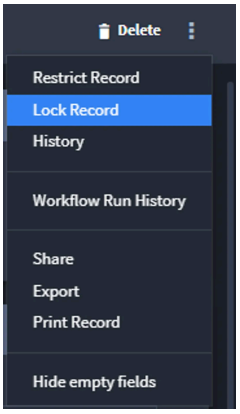


Locked records are also indicated in the report-view-of-records (Default Report) interface.



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

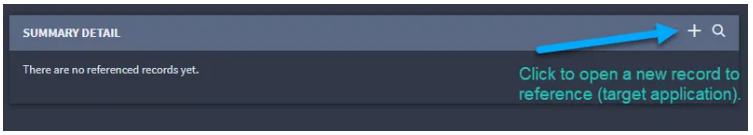
You can select which field(s) to reference from within a record. A reference field on a record, whether single-select, multi-select, or grid, is accompanied by a New and Lookup button adjacent to the reference field.

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.

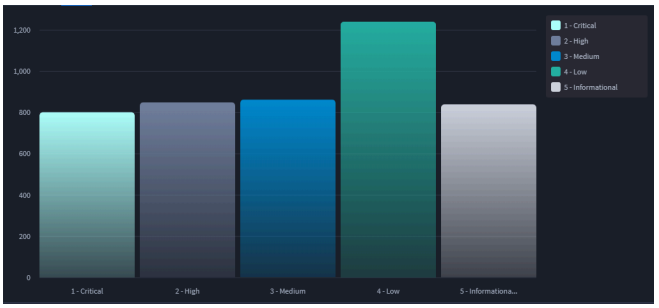
2.1.1.5.4.2. Chart Types

This topic contains chart type examples.

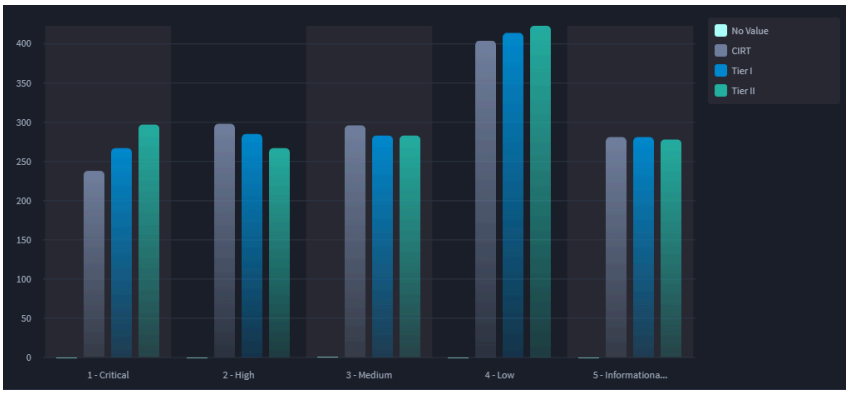
Bar Charts

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

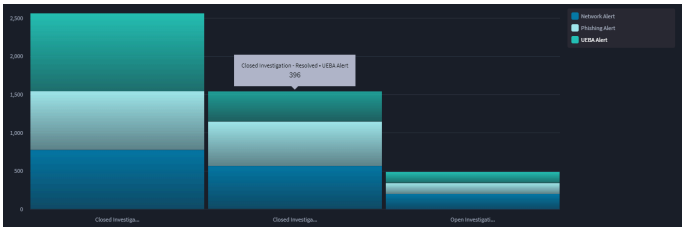
Vertical Bar



Grouped Vertical Bar



Stacked Vertical Bar



100% Vertical Bar

2.1.2.1.1. Getting Started

2.1.2.1.1.1. Playbooks Overview

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**](../03-Using Actions/03-Flows/02-02-69-Flows.md).

Playbook Data Model

On the canvas, Turbine stores playbook information in **two layers**. Understanding both layers explains what **Save** updates and why some changes appear only after you save from the canvas editor.

Layer	What you see in the UI	What is stored	What runs
Playbook (container)	Playbook name, description, canvas layout, flow order, annotations	A **builder playbook** record that lists which flows belong to this playbook	Does not run by itself; groups flows
Flow	One trigger and its	A **playbook** (flow) document:	This is what

2.1.2.1.3.1.2. Playbook Button Triggers

Playbook Button Triggers create interactive buttons in application records that, when clicked, execute a playbook with the current record's data. Unlike automatic triggers (like Record Event or Schedule triggers), button triggers give users control over when to execute a playbook, making them ideal for manual workflows, approvals, or user-initiated actions.

Key Benefits

- **Manual Control:** Users decide when to execute the playbook by clicking the button
- **Record Context:** Full access to the current record's data when the playbook runs
- **Seamless Integration:** Buttons appear directly in the application record interface
- **Flexible Configuration:** Create buttons from the playbook canvas or application builder
- **Reusable Buttons:** Multiple playbooks can use the same button, or create new buttons as needed
- **No Flow Rebuilds:** Buttons can be added or modified without requiring reconstruction of the entire flow

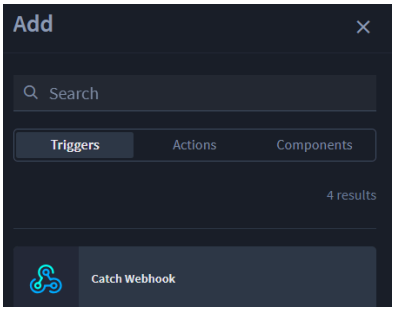
How to Create Playbook Button Triggers

You can create a Playbook Button using two methods:

- From the playbook
- From the application

Method 1: Create Playbook Button from a Playbook

1. From the **Add** section, drag and drop **Playbook Button** to the canvas.
2. Click on the added Playbook Button to view the **Trigger** configuration panel.



2.1.2.1.4.3. Emit Event Native Action

The **Emit Event** native action enables the emission of events that can trigger other playbooks through Flow Event Triggers. The Emit Event action sends events with specific data properties that are passed to the Flow Event Trigger for use in the triggered playbook. This allows actions in one playbook to dynamically trigger actions in another playbook.

Overview

The Emit Event action requires two inputs:

- **sensorName**: The name of the flow event sensor to emit (required). This must match an existing Flow Event sensor.
- **eventData**: The data object to include in the emitted event (required). The structure of this data should match the schema defined for the flow event.

When the action executes successfully, it publishes a SensorEvent with type `flow` and action `emit`, which can be received by Flow Event Triggers in other playbooks. The action uses an "at most once" execution guarantee to prevent duplicate event emissions.

Important: Turbine limits event chaining (event -> playbook -> emit event -> playbook) to 10 levels by default.

Adding and Configuring the Emit Event Native Action

1. Drag and drop the **Emit Event** native action from the native action panel into the playbook.
2. Click on the **Emit Event** action to open its configuration panel.
3. In the **Info** section:
    - **Title**: Provide a unique and descriptive title for the action (for example: "Emit Event").
    - **Key**: Ensure the key is auto-generated or manually configured to match the naming conventions (alphanumeric and underscores only).
    - **Description** (Optional): Add a description for clarity about the action's purpose.
4. Click the **Configure** button under the **Flow Event Data** section.

2.1.2.1.4.8. Using Variables in Playbooks

The **Variables** feature in Swimlane Turbine allows orchestrators to collect, store, and use data across various actions within playbooks. By leveraging the **Create Variables** and **Update Variables** native actions, orchestrators can define and modify variables without writing any code. This enables greater flexibility and ease of use in complex workflows.

Overview

Variables in Swimlane Turbine serve as temporary data storage that can be dynamically manipulated throughout the execution of a playbook. This allows playbooks to make decisions, manage data, and pass information between actions seamlessly.

Key Benefits

- **Dynamic Data Handling:** Easily collect, modify, and utilize data throughout the workflow.
- **Code-Free Configuration:** Create and update variables without the need for scripting.
- **Improved Flexibility:** Enable complex logic and decision-making capabilities by dynamically adjusting the flow of actions.
- **Seamless Integration:** Variables can be used across different actions, ensuring smooth data flow and continuity within playbooks.
- **Promotable Variables:** Promote variables for accessibility across multiple playbooks, enhancing reusability and reducing redundancy.

Creating Variables

The **Create Variables** native action enables orchestrators to generate new variables that can be applied across downstream actions. **Important:** You cannot create a variable that already exists. If you attempt to create a variable with a name that already exists, the action will fail. Use the **Update Variables** action to modify existing variables instead.

Here is how to use it:

1. Start by adding a trigger or existing playbook action to ensure upstream data is available for the variables.

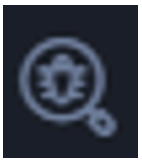
2.1.2.1.5.1. Test Console

The Turbine Canvas includes a Test Console within the playbook canvas, allowing end-to-end testing of entire playbooks and their trigger flows. This console displays each automation step to indicate whether a run succeeds or fails. You can test using real data or custom test data, offering a flexible environment for playbook verification.

> The Test Console is used for testing entire playbooks. For testing individual actions, use the **Test** tab within the action configuration dialog.

Accessing the Test Console

To open the Test Console, click the **Test** icon on the playbook canvas.



Understanding the Test Console Layout

After opening the Test Console, you'll see the flow being tested on the left side. This layout provides a clear view of each automation step. To the right, there are three main tabs:

- **Runs**: Displays the date and timestamp for each test run, indicating whether it was successful or failed. You can expand each run to view detailed information about trigger data, action data, and published results.
- **Test Data**: Allows input modification for custom testing. The available options depend on the trigger type:
  - **Webhook Triggers**: Select from past events or enter custom JSON
  - **Record Event Triggers**: Select from historical records
  - **Flow Event Triggers**: Enter custom JSON
  - **Schedule Triggers**: No test data required (test runs immediately)
- **YAML**: (Available when feature flag is enabled) Displays the playbook definition in YAML format.

Icons next to each run provide a quick status:

2.1.2.2.2. Components

As an orchestrator, Turbine offers two helpful options for building components. You can either use a Swimlane-built component, if it meets your needs, to save time and effort to achieve a desired outcome, or you can build and customize your own reusable component.

Customizing components that exist in your component library saves you time recreating or duplicating work, copying a component and modifying as needed, and provides desired-outcome flexibility.

Components, also known as vendor interaction components (VICs), focus on the intent of the vendor action. Vendor APIs send data in differing formats. That data needs to be in common data formats for best practices. Use components to set an intent, then configure vendor-specific details.

Turbine components focus on:

- Ingestion
- Enrichment

Why Use Ingestion Components

Turbine ingestion components get data from third-party tools and transform that data into appropriate Open Cybersecurity Schema Framework (OSCF)/Turbine Extendable DataSchema (TEDS) objects.

Using ingestion components:

- Provides preconfigured intents for your playbook framework to reduce time building
- Allows mass data ingestion
- Uses ingested data downstream in the playbook and/or the promoted results for use outside the playbook

Why Use Enrichment Components

Turbine enrichment components ingest data from third-party tools and transform data into appropriate OSCF/TEDS objects to improve incident response investigations for threat hunting.

2.1.2.3.2.1. Alerts

The **Alerts** section in Swimlane Turbine provides visibility into alert notifications generated when playbook runs fail. It complements the configuration of orchestration alerts by showing administrators the history of triggered alerts within a selected timeframe.

Overview

The Alerts page displays a table of alerts, including:

- **Alert** – The type of alert generated.
- **Time of Alert** – When the alert was triggered.
- **Details** – Additional information about the failure or notification.

If no alerts are available for the selected time range, the table will display an empty state message: *"No alerts available for the selected time range."*

Time Range Filtering

You can filter alerts by selecting a time window (for example, Last 24 Hours, Last 7 Days, or Last 30 Days). This allows administrators to focus on recent issues or review longer-term trends in orchestration failures.

Configuring Orchestration Alerts

To enable and configure alerts:

1. In the **Admin Panel**, navigate to **Settings > Tenant-specific > Orchestration Alerts**.
2. Enable the **Send via Email** option by selecting the checkbox.
3. Once enabled, a **Recipients** field will appear. Here you can specify:
    - **Users** – Directly assign alerts to specific Swimlane users.
    - **Groups** – Assign alerts to Swimlane groups to ensure multiple team members are informed.
    - **Custom Email IDs** – Add external addresses if notifications need to go to recipients outside of Swimlane.

2.1.2.4.1.4.2. Promote Playbook Inputs to Outputs

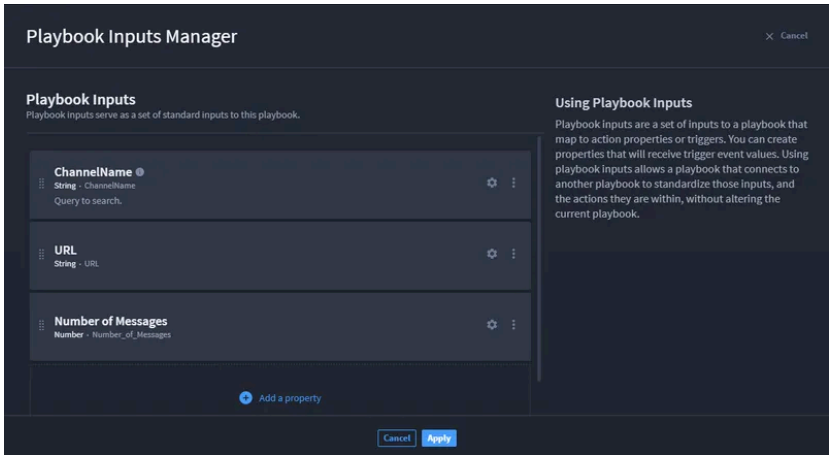
Once you create your playbook inputs or configure action inputs, you can promote the inputs to outputs, then map the outputs to applications. See **Promote Action Outputs** for more information about promoting from the action-level.

> To promote and map playbook inputs to outputs, ensure that you complete the steps to create playbook inputs and/or configure action inputs. See **Create and Edit Playbook Inputs** and/or **Create Actions and Configure Inputs** for details.

Promote Playbook Inputs to Outputs Example

This example has two parts. First, let's add a new property in the Playbook Inputs Manager. This playbook example has a Slack and a VirusTotal action. The inputs have been configured. To add a new property to inputs:

1. From your playbook, click **Playbook Inputs**.
   The **Playbook Inputs Manager** opens and existing input values display. Let's add another input, then promote.
2. On **Playbook Inputs Manager**, click **Add a new property**.
3. From the drop-down menu, select a property.
   Let's add a number value.
4. Click **Number**.
5. On **Property Configuration**, configure the property.
   This example uses **Number of Messages**.



2.1.2.4.1.8.3. Classic Discovered Outputs and Testing

For every action, there are outputs. Basically, when an action executes (and does not fail), the output data returns a minimum base of property types, which you can apply as inputs for other actions within your playbook, or promote to use outside of the current playbook.

Discovered Outputs

Also! There are discovered outputs. Based on what action or connector you use, the results may vary. So in addition to the base of property types, and depending on the inputs, action outputs may return additional properties. These are the discovered outputs.

What if you want to test your action to see the possible outputs? This can be done at the action-level. Let's take a look at the how to test the actions/connectors that Turbine supports.

Action Testing

Let's test the action that you created! Testing a single action enables better debugging, helps you detect errors in code earlier, saves time, and lets you test inputs while playbook building.

To test connectors/actions, follow these steps:

1. Select your action and click **Configure**.
   The Request, Outputs, and Test tabs are available to configure.
2. From the Test tab, enter your inputs and configure the rest of the action as needed.
3. After entering configuring your request, click **Test** to the right.

The Result pane below provides the raw JSON data that you don't need to worry about! Turbine turns that difficult-to-read JSONata into an easy-to-read format under the **Discovered Outputs** tab. The discovered outputs are combined with the base (original) outputs. All of the known outputs display with check marks. The remaining outputs from the request will be unchecked, and you can select one or all of them and click Add those selected to outputs.

To view the added outputs, navigate to the Outputs tab. All of the outputs that were custom (that you just added) display with the option to delete. You'll notice that you cannot delete

2.1.2.4.1.9.4. Classic Record Actions

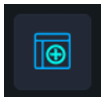
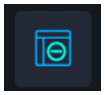
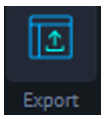
The Create Record, Update/Create Record, Search Records, and Delete Record native actions describe the functions for creating and maintaining data used in Turbine application records. These actions together are also commonly referred to as CRUD.

Record Actions in Playbooks

To access these native actions, follow these steps:

From the playbook builder, **ACTION** panel, and drag and drop one of the desired Record actions.

The following table shows the action and associated icon.

Icon	Record Action
	Create Record
	Update/Create Record aka Upsert
	Search Record
	Delete Record
	Export Record

The Create and Update/Create Record actions have the ability to configure inputs, outputs, and restrictions to your record.

Note: Restrictions can be modified on the record as well.

Restrictions and Outputs

Create and Update/Create Record actions have a Restrictions tab where you can restrict application records to specific groups and/or users.

If you navigate to the Restrictions tab on a Create action, and you have not configured any

2.1.2.4.1.10.1. Nested Playbook Triggers

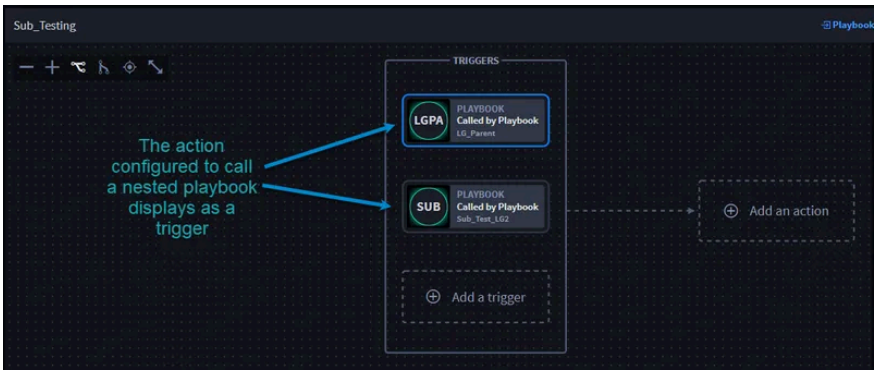
You can create parent playbooks as triggers in nested playbooks. When an action is configured to call a nested playbook, the nested playbook will show that parent playbook as a trigger.

> Review **Nested Playbooks** to understand the relationship of a parent and nested playbook before continuing.

> You do not need to select a trigger before adding and configuring actions and conditions.

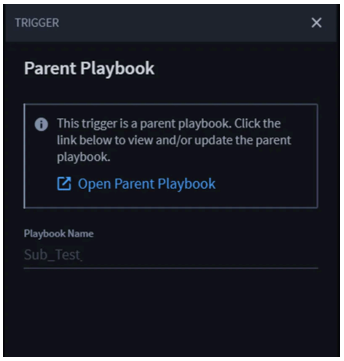
Playbook Button Triggers

> If you have more than one playbook as a trigger, click the desired playbook trigger, which will be visible with a blue highlight box around the trigger.



The TRIGGER panel displays. To view the list of parent playbooks and triggers:

1. From TRIGGER, click **Open Parent Playbook**.



2.1.2.5.1.1. Export Swimlane Solution Packages

Export Swimlane Solution Packages (SSPs, file extension `.ssp`) to transfer playbooks, applications, applets, builder components, builder solutions, and their dependencies to another Turbine instance or to the Content Library.

> Exporting SSPs requires the **System Administrator** role.

What Is Included in an Export

Included	Excluded
Root application, applet, playbook, builder component, or builder solution	Application history and record history
Associated applications and applets	Records
Workflows for root and associated entities	Secure credentials and key store values (add after import)
Reports, workspaces, dashboards, and export templates	Secure values in asset configurations (reconfigure after import)
Playbooks, including parent and nested playbooks	Connectors (maintain separately)
Orchestration tasks, ingestion rules, sensors, and correlation rules	
Assets, plugins, and Dynamic Orchestration assets	
Builder components, builder solutions, and builder intents	
Mapped inputs to key store values (keys removed; add back after import)	

Export an Application or Applet SSP

2.1.3.2.3. Create the Layout

Now that you have defined your application's foundation, you build out the format of the application. You organize your application in sections, tabs, by playbook, or with HTML.

The layout of your application consists of fields and layout objects. Drag and drop the fields and layout objects to the Form Layout area of Application Builder.

Any of the layout objects can be re-sized to a percentage of the page.

Available Layout Objects

Layout Object	Description
Tab	Groups multiple sections in the same area. Each tab can have it's own layout, which can contain additional sections and tab elements.
Section	Groups multiple fields and layout elements into a single area. Sections can contain additional sections and tab elements and can be collapsed or expanded in the record.
HTML	Displays HTML in the record.
Playbook	Adds a button to the record form that allows you to run a playbook from the records page. You must have already defined the playbook that will run in order to set up a playbook from a record.

Creating Tab Layouts

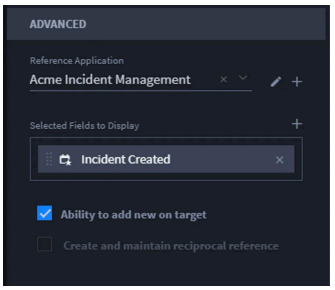
To create tab layouts:

1. From the FORM LAYOUT section of the Application Builder page, select **Tabs**, and then drag and drop it into the Form Layout.

2.1.3.2.6.7. Reference

Use a reference field to add a reference to another record, either from the current application or from another application.

For example, consider that you have an application called Address with these fields: Address, Zip Code, City, Country; and another application called Employee with these fields: First Name, Last Name. A reference field in the Employee application called "Home Address" could link to the Address application and contain the fields Address and Zip Code. When creating a record in the Employee application, a pre-existing Address record can be referenced, or a new one can be created and referenced on the spot.



When creating a reference field, you can choose from these reference types:

Field Type	Description
Single-select	Use to reference a single record.
Multi-select	Use to reference multiple records.
Grid	Use to reference multiple records with additional, user-specified, field details.

Create Reference Fields

To create reference fields:

From Application Builder's Field Types, click **Reference** and then select which Reference Type you want to create. Drag the reference field to the Form Layout and drop it in the layout area, or within a Tab or Section layout object.

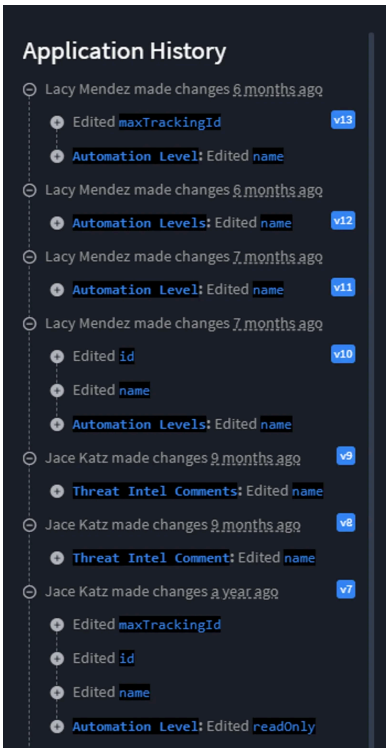
Access the field's properties and complete the following fields as needed:

2.1.3.3.5. View Revision History - Applet

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.

2.1.3.4. Widgets

2.1.3.5.2. Advanced Workflow Topics

Workflow Execution

Execution Process

Workflow execution follows this process:

1. **Initialization Phase**
   - System loads the workflow configuration
   - Initializes tracking for workflow execution
   - Creates a unique workflow run identifier (`CurrentWorkflowRunId`)
   - Increments workflow run sequence number
   - Skips execution if workflow is empty or default (`IsDefaultWorkflow()`)
2. **Recursive Stage Processing Phase**
   - Processes each stage in the workflow tree recursively (`recursiveProcess`)
   - For each stage:
     - Checks if stage is disabled (skips if disabled)
     - Evaluates all conditions in the stage sequentially
     - Determines stage validity based on:
       - Condition evaluation results (AND/OR logic)
       - Parent stage validity (`parentValid` parameter)
       - Stage disabled state
     - Queues actions for execution:
       - **Regular actions**: When conditions are met (`valid && parentValid`) and stage was not previously executed
       - **Inverse actions**: When conditions are no longer met (`stage.executed` becomes false) for reversion
3. **Condition Evaluation Phase**
   - Each condition evaluates a field value against criteria
   - Conditions use operators: equals, doesNotEqual, contains, greaterThan, etc.

2.1.3.6.1.4. Lookups

ADDRESS

Description: Returns the address of a cell in a worksheet given a specified row and column numbers.

Example:

ADDRESS(row_num, column_num, [abs_num], [a1]. [sheet_text])

Choose

Description: Returns the value from a range of values on a specific index.

Example:

Choose(index, valuearray)

Column

Description: Returns the column index of the provided column in range.

Example:

Column(range)

ERROR.TYPE

Description: Returns an integer for the given error value that denotes the type of error given

Example:

ERROR.TYPE(value)

Exact

Description: Compares two values ignoring the styles and returns the boolean value as true or false.

Example:

Exact(value1, value2)

2.1.4.4. AI-Friendly Component Best Practices

Use this guidance when you create, configure, and document components so Hero AI and AI SOC can select them reliably, map inputs correctly, and interpret outputs.

At a Glance

If you need to...	Focus on...
Help Hero AI pick the right tool	Clear **single purpose**, **name**, and **description**
Improve mapping and run quality	**Well-defined inputs and outputs**, **required fields**, and **flat schemas**
Help the model decide what to do next	**Explicit result messaging** and **clear errors**
Support AI SOC plans and dynamic mapping	**AI SOC specifics**

See also: How Hero AI Executes Components, **Components** (Hero AI visibility), **Create and Modify Components with Hero AI**, **AI Agents in Orchestration**. For **Hero AI native action** playbook prompts (failure paths, tool limits, non-deterministic output), see **Hero AI Native Action**.

Clear Single Purpose

The component should do one thing that is easy to describe. That one thing can still be complex (for example, quarantine an endpoint on the network), but it must be a single, understandable outcome.

Do not build components that do two or more separate jobs. Split separate tasks into separate components.

Clear Descriptive Name

The name should state what the component does. Avoid vague titles. If the component targets a specific product (for example, creating a firewall rule in Check Point), include that in the name, such as **Add Check Point firewall rule**.

2.1.5.3.5. Tenants

Use the **TENANTS** tab to view and manage the list of tenants. From the **TENANTS** page, you can perform the following actions:

- Rename a tenant by clicking the button and selecting **Rename**.
- Add a new tenant by selecting the **Add tenant** button.
- Search for tenants by using the **Search by tenant name** field.
- Edit a tenant's identifier (color or image) by selecting **Edit** from the **More options** menu (**...**).

Tenant Identifier Customization

Each tenant can be assigned a visual identifier to support quick recognition across the user interface. Identifiers include the following:

- **Color:** A random color is automatically assigned to existing and newly created tenants. You can change the color to align with a theme or to help group tenants by category. Specify a HEX value, such as `#b8eafe`.
- Browse for your logo file or drag and drop it into the designated field.
    - **Supported File Formats:** SVG and PNG
    - **Minimum Resolution:** 512x512 pixels
    - **Recommendation:** Use square logos with a transparent background for best results.

Tenant identifiers appear in the following areas:

- The tenant switcher in the top navigation bar.
- The **Identifier** column in the **TENANTS** table.

Tenant Table Columns

The **TENANTS** table contains the following columns:

- **Identifier:** Displays the selected color or uploaded icon.
- **Name:** The name of the tenant. This column supports alphabetical sorting.

2.1.5.4.1.2. Sessions and Security

Sessions & Security settings are only available to Turbine administrators. Use these settings to configure user session management, password policies, and authentication methods for your account.

This section includes the following topics:

- **Setting Up Session Timeout Parameters**: Configure how long user sessions remain active before requiring re-authentication
- **Setting Up Security Parameters**: Configure password policies, complexity requirements, and failed login attempt limits
- **Enable Two-Factor Authentication**: Add an extra layer of security by requiring a second authentication factor
- **Enable SAML for SSO**: Configure single sign-on using Security Assertion Markup Language (SAML)
- **Global Logout Behavior**: Control whether users are logged out of all devices or just the current session when they log out

2.1.5.4.1.3. Directory Services

Swimlane Turbine integrates with two Directory Service types: Microsoft's Active Directory (AD) and Open LDAP. By integrating with Directory Services, administrators can streamline user management and ensure consistent authentication across their organization.

Use Cases and Benefits

Many Turbine administrators leverage Directory Services to:

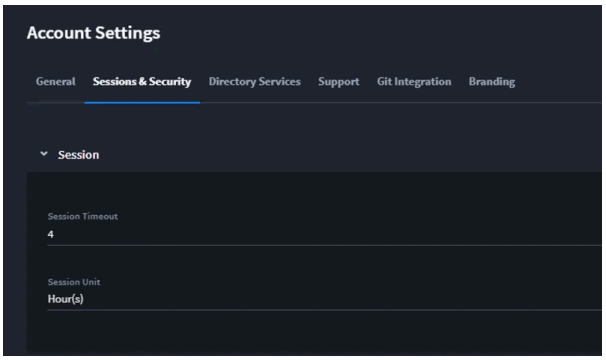
- Enable SOC Engineers and Analysts to log in to Turbine with previously established directory credentials.
- Automate user and group management, reducing administrative overhead for large teams.
- Increase security by centralizing authentication and maintaining compliance with

2.1.5.5.4. Setting Up Session Timeout Parameters

As an administrator, you control how long a Turbine session can remain idle before the user must re-authenticate or log out and log back in. The timeout is based on **inactivity**, not total time signed in. User activity such as mouse movement, keyboard input, or touch resets the idle timer.

To set up session timeout parameters:

1. From the **Sessions & Security** tab, click **>** to expand the **Session** controls.
2. Specify the length and the unit of time for the user session, then click **Save.**
3. Use the drop-down menu to specify the **Session Unit**, such as *Hour(s)* or *Minute(s)*.



> When you adjust the timeout session parameters here, users will need to log out and log back in for the updated time frame to apply. All user sessions will retain the previously-set timeout value until their current sessions expire or they log out and log back in.

Overriding a User's Session Timeout

Administrators can override a specific user's session timeout. This can be useful, for example, when using a Turbine session as a dashboard for a shared work area. It can also be helpful when you need to immediately disable a user's session.

To override a user's session timeout:

1. From the **Admin Panel**, select **Users**.
2. Choose the user whose session timeout you want to override from the Users list.

2.1.5.7.1.3. Events Dashboard

The **Events** dashboard allows you to track event activity across tenants. The following metrics are displayed:

- **Total Events:** Displays the total number of events for the selected date range, tenants, and event types.
- **Average Events per Day:** Displays the average number of events per day based on the selected filter. For example, if the last 30 days Date Range filter is applied, the dashboard calculates the average by dividing the total event count by 30.
- **Events Graph:** A graph visualizing event trends over the selected date range:
  - The horizontal axis shows the dates.
  - The vertical axis shows the number of events.
- **Events Table:** The table provides details such as:
  - Tenant
  - Event Type
  - Events
  - Triggered Playbooks
- You can sort the table by clicking on the column headers.

Filters

You can configure the Events dashboard using the following filters:

- **Date Range**: Filter event data based on a specific time period. Only data from the **last 90 days** is available. Options include:
  - Today
  - Last 7 days
  - Last 30 days (default)
  - Last 90 days
  - Custom Date Range
- **Tenant**: Select tenants to filter events.

2.1.5.8.2.3. Heartbeat and Shutdowns

Remote agents have heartbeats back to the primary system, displayed as the “Last Report Date”. If a remote agent encounters an issue, it will attempt to return a health reason to display (for instance, if it receives a SIGTERM from the OS). However, if no reason is received, the report will simply show “Agent is unresponsive”.

2.1.5.8.3. Assigning Pools to Actions

About Pools

- Pools determine which agents execute specific playbook actions.
- Agents can belong to multiple pools; pools can contain multiple agents.

Examples

- Assign agents in different networks to separate pools for resource isolation.
- Assign agents across zones to the same pool for high availability.

Assigning a Pool in a Playbook Action

1. In your playbook action, click **Show advanced settings**.
2. The default is $default. Use the dropdown to select your desired pool.
3. Save your action.

Important: Transformation and Python actions do not function with remote agents.

2.1.5.8.4. Troubleshooting

2.1.6.1. Working with Interfaces

Overview

This section of the Turbine User Guide describes the interfaces available in Turbine Solutions. Interfaces are standard data formats that enable components to work together seamlessly. Use interfaces to standardize data transformation across security operations workflows.

When you apply an interface to a component, it automatically configures the component's input and output data structures. This standardization allows you to easily swap components in your playbooks without manual re-configuration, as long as they use the same interface.

This guide covers:

- What interfaces are and how they work
- How to use interfaces when building components
- Available interface catalogs for SOC, AI SOC, and VRM workflows
- Best practices for working with interfaces

Interface catalogs

Full interface contracts for each solution area live in these guides:

- **SOC Interfaces** — 22 Classic SOC interface contracts
- **AI SOC Interfaces** — 4 AI SOC interface contracts
- **VRM Interfaces** — 6 VRM interface contracts

For complete data model field definitions, see **Turbine Schema Reference (Classic SOC)**, **Turbine Schema Reference (AI SOC)**, and **Turbine Schema Reference (VRM)**.

What Are Interfaces?

An **interface** defines the data structure that a component expects to receive (inputs) and the data structure it produces (outputs). Think of it as a standard template that ensures components can work together seamlessly.

Key Concepts

3.1.1. Accessibility Conformance Report

- **Name of Product/Version:** Swimlane Turbine Cloud 25.3.0
- **Report Date:** October 2025
- **Product Description:** Swimlane Turbine is a low-code security automation platform. With Turbine you can prioritize alerts, re-mediate threats and improve your operational performance.
- **Contact Information:** info@swimlane.com

> Turbine is a web-only application.

This report covers the degree of conformance for the following accessibility standard/guidelines:

Standard/Guideline	Included in Report
Web Content Accessibility Guidelines 2.2	Level A (Yes) Level AA (Yes) Level AAA (Yes)

Terms

The terms used in the Conformance Level information are defined as follows:

- **Supports:** The functionality of the product has at least one method that meets the criterion without known defects or meets with equivalent facilitation.
- **Partially Supports:** Some functionality of the product does not meet the criterion.
- **Does Not Support:** The majority of product functionality does not meet the criterion.
- **Not Applicable:** The criterion is not relevant to the product.

Success Criteria, Level A

Criteria	Conformance level	Remarks and explanations
1.1.1 / Non-	Parti	Some images that contain text are missing `alt` tags. Some icon

3.2.9. Monitoring Script for Remote-Agent Host Health

This article provides a script that customers can use to collect system health metrics from the server running the Turbine remote-agent containers. The goal is to capture key data points (DNS resolution, CPU, memory, I/O, network performance, etc.) at regular intervals so that, in the event of a container restart or connectivity issue, system conditions at the time can be reviewed.

This is particularly useful in environments where connectivity to the Turbine Cloud instance may be impacted by DNS timeouts, network quality issues, or resource exhaustion.

This solution has been tested on commonly used Linux distributions, including:

- **RHEL 7 / 8 / 9**
- **Ubuntu 18.04 / 20.04 / 22.04**

What the script collects

- Running containers (docker ps)
- Container IPs
- Uptime
- Memory usage
- CPU load (top processes)
- Disk I/O (iostat)
- Disk usage
- DNS resolution
- Network response times using curl

Prerequisites

- Docker/Podman must be installed and running
- Optional but recommended: iostat via sysstat package
  Install the sysstat package:

3.3.1.3. Configure String Array Condition Expressions

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

Scenario

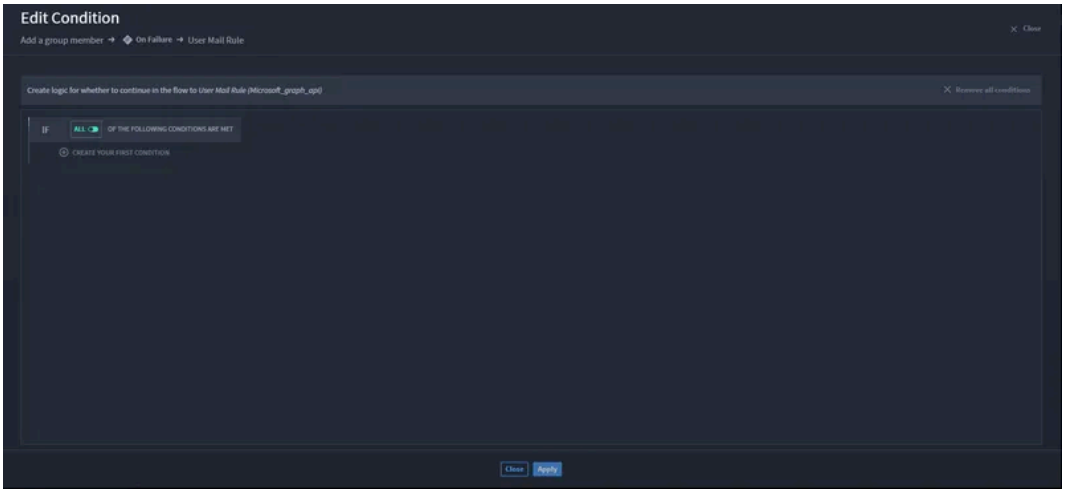
Alex wants to configure a string array in a conditional expression using the output of a JSONata action. Alex is ready to begin. He starts by adding and configuring a JSONata connector with an **On Success** action flow, and now he wants to add a string array conditional expression.

To add a condition, click the **Add condition** icon.

> Once you click the action flow, it turns blue and the FLOW panel displays to the right

The Edit Condition window opens.

Click **CREATE YOUR FIRST CONDITION**.



The available action/playbook properties displays. Click the property to expand the available input types.

Click **String Array** property.

> If available, you can select more than one String Array.

3.3.3.3. Action Retries

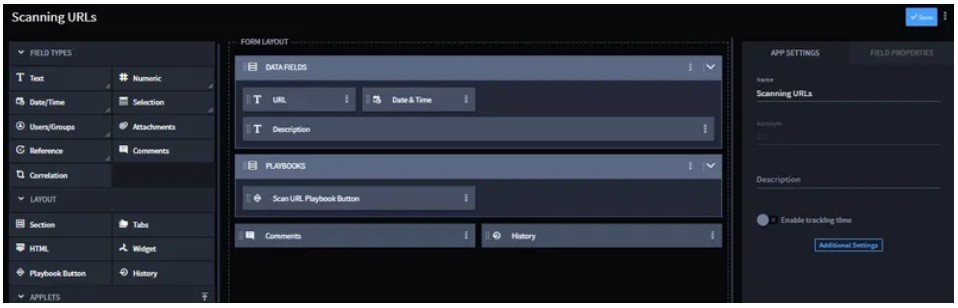
Scenario

Alex works in a Security Operations Center (SOC). She is responsible for discovering and reporting malicious URLs. She receives a report of URLs each morning. She looks up each of the URLs, and then uses the resulting scans to report which URLs are most severe. This takes up most of Alex's morning each work day.

Alex wants to use Turbine to simplify and automate this process. She has already ensured that her version of Turbine has downloaded and configured the Urlscan connector from Swimlane Content and created/saved a Scan URLs playbook, and reviewed the report to determine which data she wants to use to feed her Swimlane application(s) and playbook.

Alex can resolve this scenario with the following solution:

Build an application that includes data fields and a playbook button that serves as a record action trigger.



Alex created the **Scanning URLs** application with the following data fields: **URL**, **Date & Time**, and **Description**, and added the **Playbook Button** to link her **Scan URLs** playbook that she already created.

With the application complete, Alex can return to the **Scan URLs** playbook to configure her data.

> The playbook trigger is the Record Action (via the application playbook button).

Add the **Submit** connector that will submit URLs for analysis.
Alex is ready to configure the connector by adding the URLscan asset, which she preconfigured with the needed URL and API Key data.

3.3.6.2. Webhook Discovered Outputs

Scenario

Eli is an orchestrator who wants to test and preview outputs from a webhook event before configuring and saving the rest of his playbook. Luckily, as an orchestrator, Eli knows that testing a webhook event is like testing **discovered output and testing**. Let's watch how he tests his webhook event!

Eli has already created his playbook and added a Webhook trigger. After giving the webhook a name and generating the URL, he's ready to configure.

1. Click **Configure**.
   The Webhook Events, Outputs, and Map to Playbook Inputs tabs are available. Eli checks the Output tabs to see the original outputs for the webhook. But he wants to test for any additional outputs that might not be being pulled. To do this, he needs to test the webhook.
2. Click **Test** next to the URL and view the Result pane at the bottom of the window.
   The results display, but Eli wants to know which outputs were discovered that were not part of the original event outputs. Since Eli already reviewed and understands what **discovered output and testing** are, he is ready to add the discovered outputs.
3. Click the check mark next to any discovered outputs to add them to the action's outputs.

Conclusion

These are now available for Eli to apply downstream and as he builds the playbook.