SWIMLANE

Swimlane (10x) User Guide

CONTENTS

1. Introduction

Welcome to the Swimlane User Guide

Swimlane is a security orchestration and automation platform. You can use Swimlane to prioritize alerts, remediate threats and improve your operational performance.

Using this Documentation

This documentation explains how to set up, configure, and use Swimlane. The documentation is categorized by roles and tasks associated with those roles.

Swimlane users work with:

- Workspaces
- Dashboards
- Cards
- Records
- Reports

All of the above are populated with data you set up with:

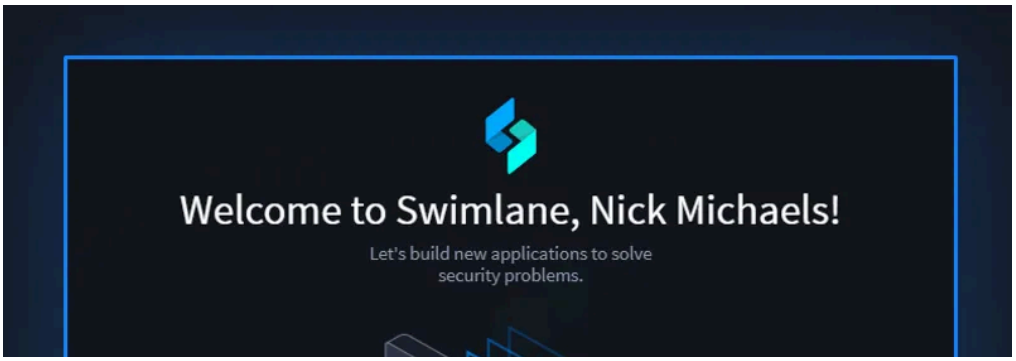
- Applications and Applets
- Records
- Tasks

Which are influenced by:

- Integrations and Assets
- Task Inputs and Outputs
- Users, Groups, and Roles

1.1. Welcome to Swimlane

Once you've installed Swimlane, you are greeted by a series of welcome screens. The screens are there to help you get started. Click the buttons on the screens to set up your user profile, start building applications, or to view the Swimlane documentation.

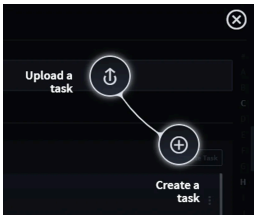


2.1.3. Create a Suspicious Emails Task

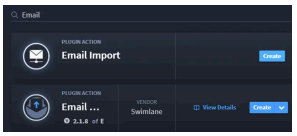
Now that you've created an application that will hold suspicious email records, you are ready to create the task that prompts Swimlane to check for suspicious emails.

To create a suspicious emails task:

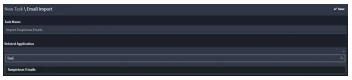
1. Access the Integrations page, then click the plus menu icon, and then click **Create a task**.



2. On Create a task, notice how Swimlane has grouped common task types by asset. Filter for Email plugin action name and then click **Create** on the Email Import plugin action.

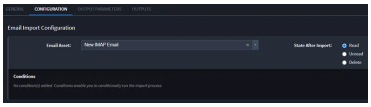


3. On page 2 of Create a Task, name your task. (We suggest naming it *Import Suspicious Emails.*) Next, from the Application drop-down field, select the application you just created in the previous topic, "Suspicious Emails," and click **Save.**



Clicking **Save** opens your task. From here you can begin to configure the task and map the task outputs.

1. On Integration/Import Suspicious Emails, click the Configuration tab.
2. Under Email Import Configuration, select the IMAP Email Asset you created in the **Create an IMAP Email Asset** topic, and ensure that you have selected the **Read** checkbox to indicate the state of the email after the task has run.



3. Next, click the Outputs tab. To begin the mapping process, click **Create/Update Records.**

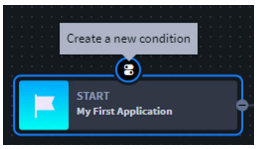
2.2.4. Add More Workflow Detail

Now you are ready to add additional detail to your application's workflow.

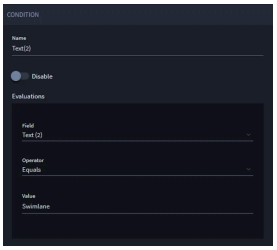
Before You Begin: From the global navigation menu, select Applications and Applets and select your application again. Select the workflow icon from the application toolbar.

To add more workflow details:

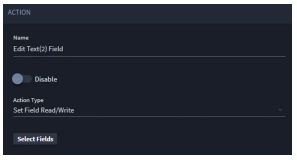
1. From within your application's workflow, select Start, or the beginning of your app's workflow, then click the icon to create a new condition.



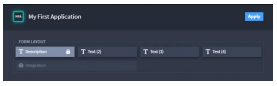
2. Edit the Condition properties as shown in the following screen, then click **Save.**



3. Next, select your new condition and click the icon for *Create a new action.*
4. On Action, specify *Set Field Read/Write* in the **Action Type** pull-down field, then click **Select Fields.**



5. Select the field that you want to set the read/write action on. Select it once to lock the field, then select it again to make it editable.



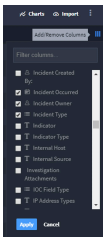
6. For example, the **Description** field above is locked. Note the lock icon.

7. The **Text (2)** field above *is* editable. Note the pencil icon.
8. Next, create another action on this condition and, on **Action Type,** select Modify Layout.

4.1. Customize the List of Records

Applications can have a large number of records. You can customize and filter your view of records according to the tasks that you need to complete.

When you first open the Default Report view of records, you'll see a Keywords and Filter taskbar and a list of records below. The records are initially listed only by Tracking ID. To adjust the columns visible, click or **Add/Remove Columns.** (You'll find it on the far-right of the Default Report page, above the list of records.)



Note: For more information on using keywords searches and filtering see, **Keyword Search and Filter**.

The columns available for you to select vary depending on how the application is built. Select the columns you want to view and click **Apply.** You must have at least one column selected.

Within the list of records, you can:

Action	How?
Resize Columns	Drag the column border to the preferred size.
Reorder Columns	Drag the column header and drop it to the desired location.
Sort Records by Column	Click the column header. Each column has three sort types, ascending, descending, and none.
Load Additional Pages of Records	Scroll to the bottom of the page and click an additional page number.
View or Edit an Individual Record	Click the record row. The record edit window opens.
Exit the Record Edit Window	Click outside of the edit window.
View or Edit Multiple Records	Click the checkbox just to the left of the record Tracking ID. Select as many records as you want to view or edit. For more information see **Bulk Modify Records**.

4.6. Records and Workflow

Workflow conditions, repeats, and actions are executed in real-time as you interact with records.

Activity on individual records catalyze workflow, for example:

- You open a new record form.
- A new record is saved for the first time.
- An existing record is opened.
- You modify an existing record.

In addition, workflow is also carried out when:

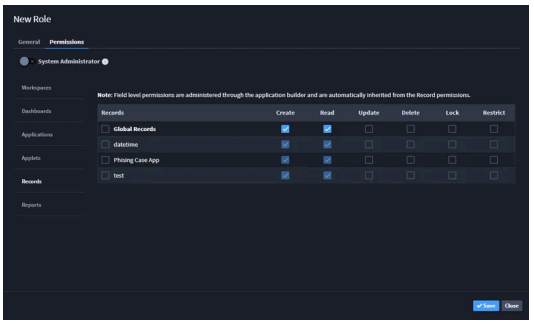
- A new record is created by an integration task.
- A new record is created by a remote client tool or script.
- An existing record is modified by either of the previous two actions.

Workflow actions are applied with the following considerations:

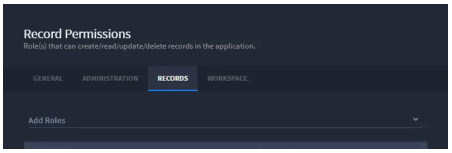
- Actions are executed in a one-pass system on each digest so as not to cause circular execution.
- Set Field Value conditions will only execute the first time the condition is met. If the condition is invalidated and then revalidated it will execute again.
- Notifications and exports are only executed on record save. This is to prevent flip-flop during editing causing multiple notifications that might not be valid.
- Integrations can be optionally setup to execute in real-time or on record save.

Required Role-Based Access Control (RBAC) Settings

For non-administrator users who want to view workflow run history, ensure that user is assigned a role that allows global record create and read permissions. You set Global permissions from USERS, GROUPS & ROLES, Roles.



Next, ensure that you assign the role to have *Read* permissions to the specific application and its workspace.



5.3.2. Share Reports by Email

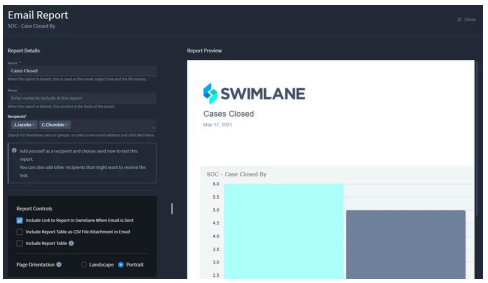
On the report page, access the ellipsis drop-down menu and select **Email** to access the Email report page.

This feature allows you to email your report to recipients. Unlike Schedule Reports, the emails sent via Email Report are sent automatically and cannot be scheduled to send at a later time. The email will only be sent one time.

1. Once in **Email Report**, under **Name**, type a name for your report. This name is visible in the Report Preview once you click out of the **Name** text box.
2. The name is the subject line of the email.
3. Under **Notes**, type any notes you would like to include on the report. These notes are visible in the Report Preview once you click out of the **Notes** text box.

Notes display as the body of the email.

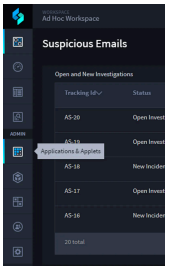
1. Under **Recipients**, search for and select the name of the Swimlane users or groups you would like to add.
2. The recipients that display in this field are other Swimlane users that have role-based access to the report. Select from those users. Know that you can also manually enter email address for people who are not Swimlane users.
3. **Note:** Name and Recipients are required to send your report.



4. Under **Report Controls**, you can chose to include a link to the report in Swimlane, to include the report table as a CSV file attachment, to include a report table, and select the page orientation of the report.
5. **Note:** When including a report table, the table displays a maximum of 25 records. If the report contains more than 25 records, you receive a message at the bottom of the report table that indicates how many total records are included.
6. **Note: Unavailable Report Controls** appears if there is a report containing a widget, or if the report does not contain a chart. This lets you know that the report will be including the report table since widgets are not supported in shared reports and the chart cannot be displayed.

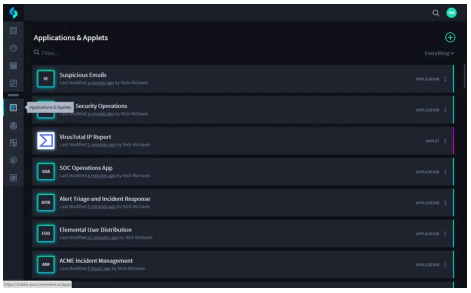
6.1. Manage Applications and Applets

You can manage your apllications and applets from the Applications and Applets home page.



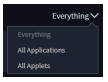
From the main page you can:

- Create new applications and applets.
- Edit existing application and applet content and settings.
- Copy applications and applets.
- Delete applications and applets.
- View applications, applets, or both.
- Import Swimlane solution packages.
- Export applications and applets as Swimlane solution packages.



You can view and modify the applications and applets you have permissions to view and modify. See **Permissions** for more information about setting user, group, application, and field permissions in Swimlane.

The list of applications and applets can be modified with the sort drop-down. You can view all applications, all applets, or view everything.



Note: The default sort order of applications and applets lists them by those most-recently accessed.

Each application and applet have associated actions according to your permissions, including:

Feature	Function
Builder	Opens Application or Applet Builder.
Workflow	Opens the workflow for the application.

6.2.3.6. 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 Field Properties tab 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.**

6.2.3.7. 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.

6.4. Applet Builder

Applet Builder is the starting point for building and managing applets. With Swimlane's Applet Builder you can:

- Create and modify fields and layouts.
- Set field level permissions and properties.
- Configure calculations.
- Access, build, and update workflow.
- Run manual validations.
- Create export templates.
- View revision history.
- Delete applets.

Applets share nearly all of the same features as applications. There are a few important differences, including the way applets handle reference fields and the way that applets configure workflow upon the addition of an applet to an application.

> Users who do not have permissions set on any applications, but have global permissions set in all other areas of Swimlane, will be able to view applications and applets in the Administrator menu. If users have global permissions for applets, they will be able to edit applets. However, since they do not have any permissions set for applications, they will see applications listed, but will be unable to edit them or otherwise modify them.

Page Components

The Applet Builder page consists of these main components:

Component	Description
Field Types	Available fields that you use to build your applet. The fields support the various input data for the records.
Form Layout	The container where you modify the arrangement of fields for your applet.
Applet Settings	Settings you assign that are relevant to the entire applet, such as Name, Description, and permissions.
Field Properties	Settings and properties that you assign for specific fields within the applet, such as permissions, size of the field in the layout, values, and whether the field is required.
Hidden Fields	A list of fields that exist within the applet and are available for reports and modification in workflow, but that are not visible in the record entry form.

You control the view of Applet Builder. Look for the small keyboard icon in the lower left corner of the applet.

6.6. Workflow

Every application in Swimlane has a workflow. Workflow is where you define how your organization processes tasks. You build the workflow like a tree, with one or more branches. Each branch is conditional, and can consist of multiple conditions and actions. In addition, you can also conditionalize repeat actions in workflow.

Activity on individual records catalyze workflow, for example:

- You open a new record form.
- A new record is saved for the first time.
- An existing record is opened.
- You modify an existing record.

In addition, workflow is also carried out when:

- A new record is created by an integration task.
- A new record is created by a remote client tool or script.
- An existing record is modified by either of the previous two actions.

Swimlane workflow consists of:

- Conditions - decision points in the workflow.
- Repeats - iterative task items within the workflow.
- Actions - singular tasks to be accomplished by workflow.

> Repeats iterate an integration action with a specific condition for each item and work exclusively with Text, Numeric, Grid, and Multi-select Reference fields.

For information on how applets handle workflow, and how workflow is updated when applets are added to applications, see **Applets and Workflow**.

To view, create, or update workflow, click the workflow icon on the Application or Applet Builder toolbar.

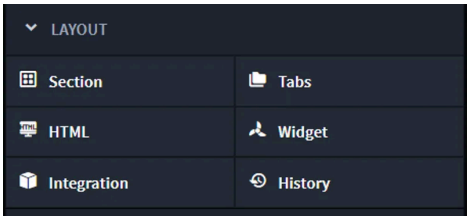
Workflow processes conditionally and can become quite complex. To zoom in to a flow, use the scroll wheel of your mouse, or use the zoom buttons in the upper left corner of the page. From this menu you can also collapse or expand all of the workflow..

Left-click, hold, and drag your mouse to navigate the branches of the workflow. You can also expand or collapse conditions within your workflow so that you can focus on a specific section. To collapse or expand a branch, click the **+** or **-** icon to the right of the condition or action.

To search for a condition or action, place your cursor in the search field and then enter a term from the name of the condition or action. Swimlane locates each instance of the term, highlights it,

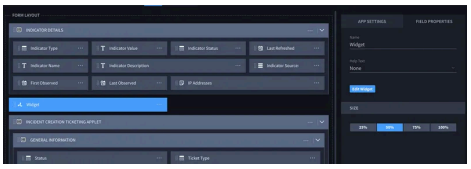
6.7.2. Record Widgets

To create record 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 record widget, select the widget in the layout. Then, in the properties sidebar, click "Edit Widget."



Record 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.

The record attribute is a javascript object, where the keys are the field's key property, and the value is the field value for that record.

Example: If an application has a text field with the key `text` and a numeric field with the key `numeric`, the record object would look like this:

```
{ text: 'text field value', numeric: 55 }
```

This object can be used in the widget's `update` method to generate the updated HTML of the widget after a change.

Widgets do not store any data; they are only used to build a UI that interacts with other data on the record. If any data needs to be stored, do so in a field on the record (either visible or hidden).

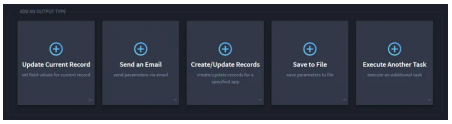
Exporting an application or applet will also export the widget as part of it, so it can be easily imported to another instance of Swimlane. This allows the record widgets to be shared and reused very easily by embedding them into an applet and exporting it. Since the widget's code is also included in the exported application or applet JSON file, Swimlane recommends that you **do not** include any sensitive information in widgets.

6.7.3. Report Widgets

You create report widgets from within the Charts feature on the Default Reports page. Click **Charts** from the Default Reports taskbar, then from **Chart Types,** select the Widgets icon.

8.4. Configure Task Output

Click the Integration, **Outputs** tab to configure how Swimlane will handle the task's output.

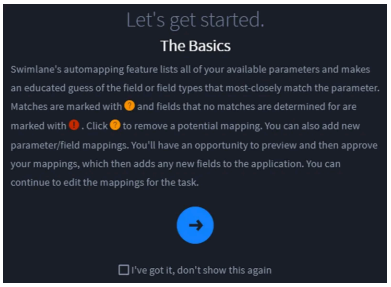


To add new output, first specify the output type:

- **Update Current Record:** Sets field values for a current record. This method, which modifies an existing record with the task's output data, must be initiated with triggers, workflow, or by manual integration. You can have one mapped per task.
- **Send an Email** - Sends parameters in an email.
- **Create / Update Records** - Creates or updates records from the data returned by running the task. If the task returns a sequence of data, you can create or update multiple data records.
- **Save to File** - Saves to specified file.
- **Execute Another Task** - Associates the task output to another task, passing the output to the next task as an input.

> Make sure that the Swimlane field and the source field type are both set up to handle the parameters for the output!

The next steps will help you map task output data and parameters to Swimlane fields. You can also specify specific parameters for the task's output, for example, attachments.



Parameter Values

You have to map, or review, parameters for output types, although mapping can be different for each configuration.

Either you, or Swimlane's automapping process can assign parameter values during the mapping. Some parameters such as Output Parameters and Run Status Parameters are available for automapping. However, Standard Output, Input Parameter, and System Output Parameters are never part of auto-mapping and will need to be mapped manually.

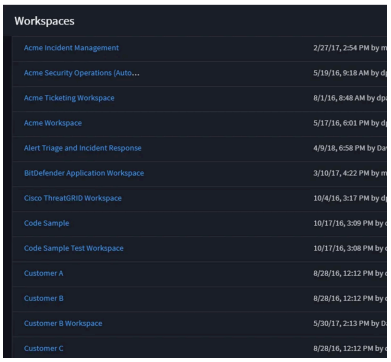
For additional information on automapping, see **Map Actions**.

Here are the parameter values that can be assigned:

9. Workspace Management

Workspace and Dashboard Management

As an administrator of a workspace, you have access to the Workspaces page.



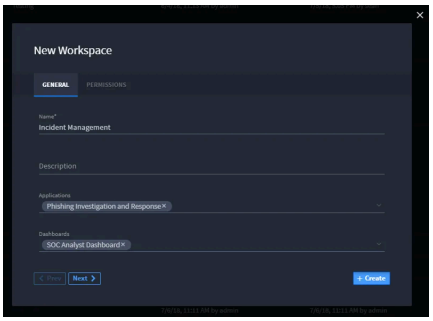
From the Workspaces page, you can:

- Create New Workspaces
- Edit Existing Workspaces
- Delete Workspaces
- Access Dashboards

9.1. Create or Edit Workspaces

You can create, edit, and delete workspaces from the Workspaces list page.

To create a new workspace, click **+New Workspace**. Define the general information about the workspace and set permissions as needed.



When you create a new workspace, you can specify the following:

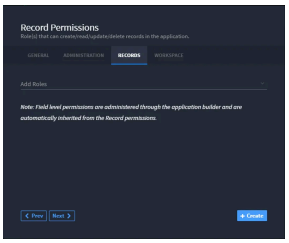
- Display Name
- Description (Optional)
- Linked Applications and Dashboards
- Manage Permissions

To edit existing workspaces, click the pencil icon and to delete existing workspaces, click the trash

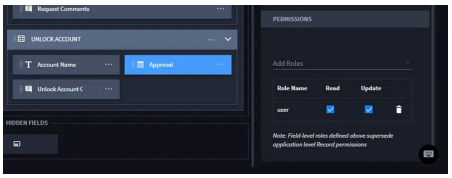
10.5. Field-level Permissions

Field-level permissions are administered through Application Builder and are automatically inherited from Record Permissions applied at the application level.

From the application settings, click Record Permissions and then select the roles you want to allow access to the application's records in **Add Roles**.



Roles can also be given permissions to individual fields within an application. From the Application Builder, select the field from the application layout and then, in the Field Editor, go to Permissions.



Roles with read permissions will be able to view but not edit the selected field. Update permissions allow the user to edit the field. Roles with no permissions set will not be able to see the selected field on the record page.

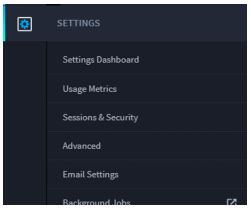
		Role-level Permissions			
		Read	Read/Modify or Reach/Modify/Create	All (specific objects)	All (global)
Field-level Permissions	None	Hidden	Field shown, value hidden	Field shown, value hidden	Modify
	Read	Read	Read	Read	Modify
	Read/ Update	Read	Modify	Modify	Modify
	Update	Hidden	Modify possible (overwrites existing), hidden after save	Modify possible (overwrites existing), hidden after save	Modify

11. Settings

Swimlane Settings

Swimlane Settings control global Swimlane settings and contain error and troubleshooting tools.

Click the Settings icon in the navigation menu to access the settings dashboard. You can also hover over the icon and select from the available options.



11.2.5.2. SAML Auto-Provisioning

Overview

Auto-provisioning is a process that automates the creation of user accounts in a system or application. When enabled, it ensures that a new user account is created automatically when a user logs in with valid credentials, such as through a Single Sign-On (SSO) service.

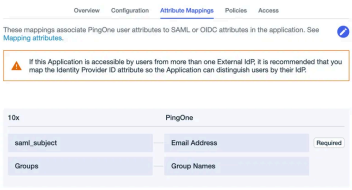
For example, in the context of SAML-based SSO, if a user attempts to log in and their credentials are authenticated successfully, the system will automatically generate a basic user account. This account may be created without any pre-assigned roles or group memberships, which can then be configured later as needed. If the user is associated to a particular group, the group is automatically synced with the user on creation.

Prerequisites

- Access to the Admin Portal.
- JumpCloud or any SSO account credentials.

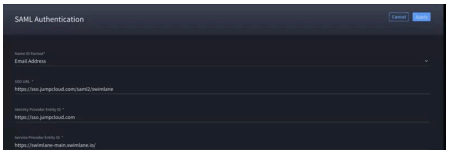
Group Synchronization with Swimlane during User Creation:

To enable automatic group synchronization in the Swimlane application during user creation via SSO, ensure that the Groups attribute is included in the attribute mapping configuration of your respective SAML identity provider (for example, Azure AD, Okta, JumpCloud, and so on). This allows Swimlane to receive and assign user group information at the time of login or provisioning. For more details, see the following image.



Enable Auto-Provisioning

1. Navigate to **Settings** > **Sessions & Security** > **Authentication**.
2. Turn on the **SAML Authentication** toggle.
3. Click **SAML Settings**.
4. Configure SAML Auto-Provisioning:
    1. Fill in all the required authentication details. Refer to SAML authentication for more information. For more information on filling the SAML Authentication dialog box refer to Enable SAML for SSO.
    2. Toggle Enable **SAML Auto Provisioning** to On.
5. Save the changes.



11.8. Logging

You can view Swimlane logs on the Swimlane Settings, Logging page.

You control the level at which logs are reported. Logging levels are progressive in the order seen in the table below. For example, if you have set the logging level to Info, you will receive the logging details for the Info level, as well as those for Debug, Warn, and Error. If you have set the logging level to Warn, you will only receive the logging details for Warn and Error.

Logging levels include:

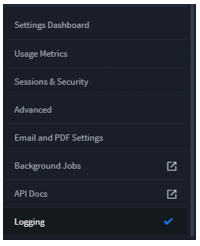
Logging Level	What the level includes:
Info	Audit information such as when and what user saved or updated an application or record and all levels below
Debug	Detailed internal system information used for problem resolution and all levels below
Warn	Warnings and all levels below
Error	Errors when something does not complete successfully

> Use the messages from logs when contacting Swimlane support regarding an error.

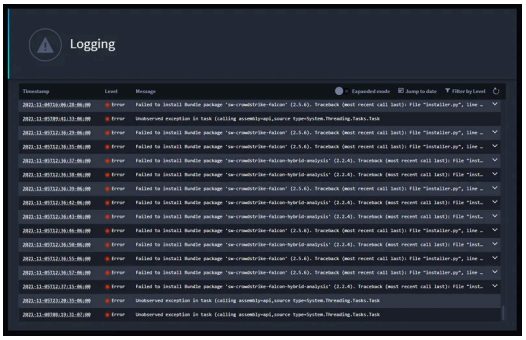
Viewing Logs

To view logs:

1. From the navigation menu, click **Logging.**



2. On Logging, the log data displays.



Log Item	Description

13.2.3. Transport Encryption

Care should be taken to ensure Swimlane traffic cannot be intercepted in transit. Swimlane web traffic should be secured in transit using TLS/SSL in IIS.

If a single Swimlane server deployment is used, the database ports of 2424, 2480, and 27017 can be blocked to restrict external database access. If MongoDB is located on a separate server, or if multiple database servers are used, consider using TLS/SSL for transport encryption of data in transit between the database servers and the Swimlane servers.

More Information

- MongoDB's TLS/SSL documentation

13.2.4. Data Encryption at Rest

Swimlane stores passwords and credential store data securely. If required, MongoDB allows for encryption of the entire Swimlane database at rest. Be aware of the performance penalty for the overhead of encrypting and decrypting the data from disk.

For more information, consult **MongoDB's Encryption at Rest**.

13.2.5. Third-Party Certificates Volume

Use the information in this topic if you need to allow Swimlane integrations and services to trust a custom certificate authority.

The third-party certificates volume can be utilized by commenting out the `volumes` key on the api and tasks containers in `docker-compose.override.yml`.

Specify the path to a local directory on the host that contains the files you wish to share with the tasks service.

On each start of the api and tasks services, run `update-ca-certificates` to import the certificates.

The api and tasks containers will need to be restarted for the changes to take affect if new certs are added.

See the following example:

```
sw_api: volumes: - /opt/swimlane/ca-certs/:/usr/local/share/ca-certificates/swimlane/ sw_tasks: volumes: - /opt/swimlane/ca-certs/:/usr/local/share/ca-certificates/swimlane/
```

13.2.6. Extend Third-Party Certificates for Python Package Installation

13.3.2. Directory Services Synchronization: Field Attribute Mappings

When configuring Swimlane's Directory Services (DS), the default values usually work fine. However, there are times when the defaults do not provide the expected results. This is typically due to the environment's DS customizations and requires an investigation into their DS tree structure to determine the correct field attributes to be selected rather than using specific defaults.

For example, one domain may use the value within *displayName* as the primary means of identifying a person, while another may use *userPrincipalName*, or *sAMAccountname*. Without querying the domain's DS schema field configurations, you might have to determine the LDAP attributes through trial and error. Instead of that time consuming activity, consider the help provided in this topic to troubleshoot this issue.

See the table below to view some of the available Active Directory fields for a user. This table also shows the associated LDAP Attribute for the corresponding AD schema field.

Common AD to LDAP Mappings for Users

	Active Directory Field	LDAP Attribute
General Tab	First name	givenName
	Initials	initials
	Last name	sn
	Display name	displayName
	Description	description
	Office	physicalDeliveryOfficeName
	Telephone number	telephoneNumber
	Telephone number (Other... button)	otherTelephone
	E-mail	mail
	Web page	wWWHomePage
	Web page (Other... button)	url
Account Tab	User logon name	userPrincipalName
	User logon name (pre-Windows 2000)	sAMAccountname
Organization Tab	Title	title
	Department	department
	Company	Company
	Manager	Manager
	Direct reports	directReports

14.4.1. REST API and the Swimlane Python Driver Library

The Swimlane platform provides a REST API with numerous endpoints that allow developers to harness and extend the platform's automation and orchestration power. As a developer, you can use any programming tool with an HTTP library or capability. You can also use the **Swimlane Python Driver** documentation, which is a client library that can streamline your efforts to implement solutions.

In general, when authoring Python scripts for Swimlane, follow these guidelines:

- When authoring scripts that will run outside of the platform, use the Swimlane Driver whenever possible and fall back on the unwrapped REST API when necessary.
- **Tip:** See **Custom Direct Requests** from the Swimlane Python Driver documentation to review the preferred method for blending these two approaches.
- When authoring scripts that will run within Swimlane, use the Task API first, and then fall back on the driver and/or REST API when necessary.

Swimlane Python Driver

This is an example of a Swimlane Python Driver script for connecting to Swimlane and creating a new record in a specified application:

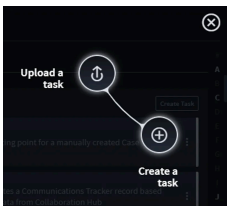
```
from swimlane import Swimlane

# Connect Swimlane client
host = 'HOST'
base_url = 'https://{0}'.format(host)
swimlane = Swimlane(base_url, 'USER', 'PASSWORD', verify_ssl=False)

# Retrieve App by name
app = swimlane.apps.get(name='Event')

# Create new Record
new_record = app.records.create(**{
    'Event Name': 'demo security event 1',
    'Event Message': 'this event is for demonstration purposes'
})
```

```
from swimlane import Swimlane

# Connect Swimlane client
host = 'HOST'
base_url = 'https://{0}'.format(host)
swimlane = Swimlane(base_url, 'USERNAME', 'PASSWORD', verify_ssl=False)

# Retrieve App by name
app = swimlane.apps.get(name='Event')

# Fetch existing record
record = app.records.get(tracking_id='EV1-4')
```

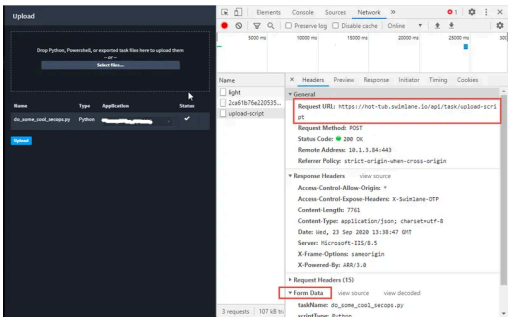
15.8. Upload a Python Script with the POST Script Endpoint

Many of the Swimlane RESTful API endpoints require JSON payloads. However, the POST /task/upload-script endpoint requires multipart/form-data.

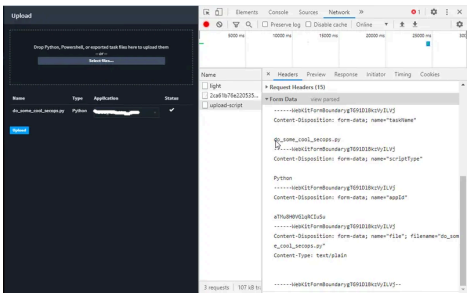
You upload JSON files from accessing **Upload a task** on the Integrations/Tasks tab.



Here is an example of a successful POST:



Here is the POST body payload that needs to be replicated in client Python code:



This Python 2.7 code sample, which relies on the Swimlane Driver for authentication and for ease of retrieving the pertinent application's ID value, illustrates how to make this POST:

```
from swimlane import Swimlane host = 'https://SWIMLANE-HOST' swimlane =
Swimlane(host, 'USER', 'PASSWORD', verify_ssl=False) app =
swimlane.apps.get(name='APPLICATION-NAME') resp = swimlane.request( 'post',
'/task/upload-script', data=[ ("taskName", "do_some_cool_secops.py"),
("scriptType", "Python"), ("appId", app.id) ], files={ "file":
open(r'C:\Swimlane\shared\do_some_cool_secops', 'rb'), "contentType": "text/plain"
} ) print str(resp.status_code) print resp.text
```

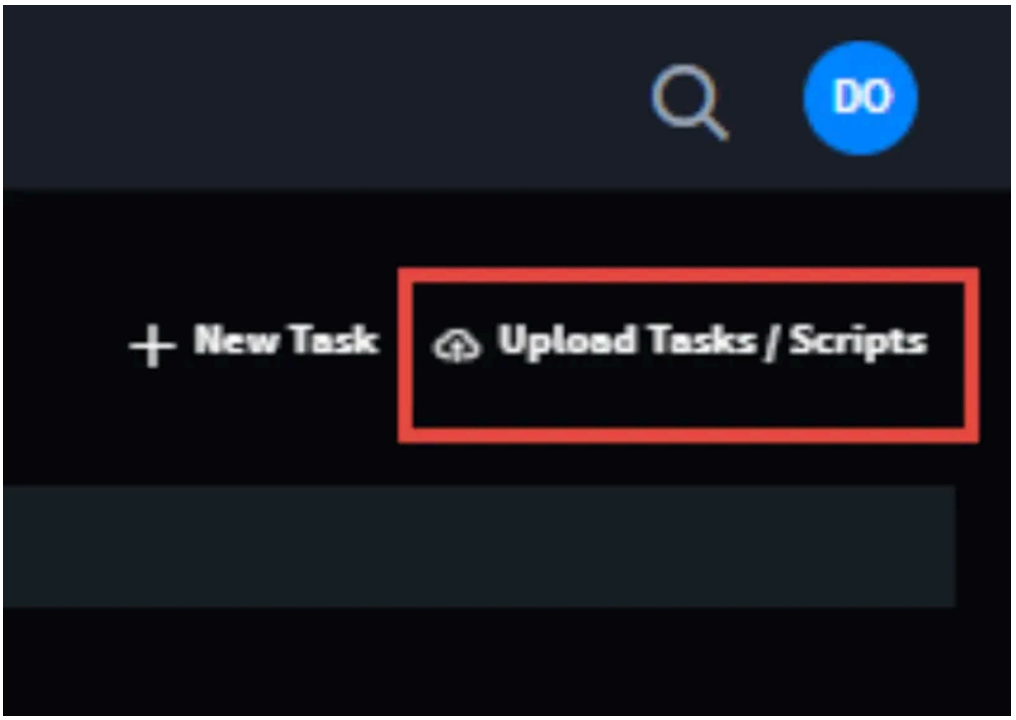
Swimlane supports Python scripts and Powershell scripts.

16. Appendices

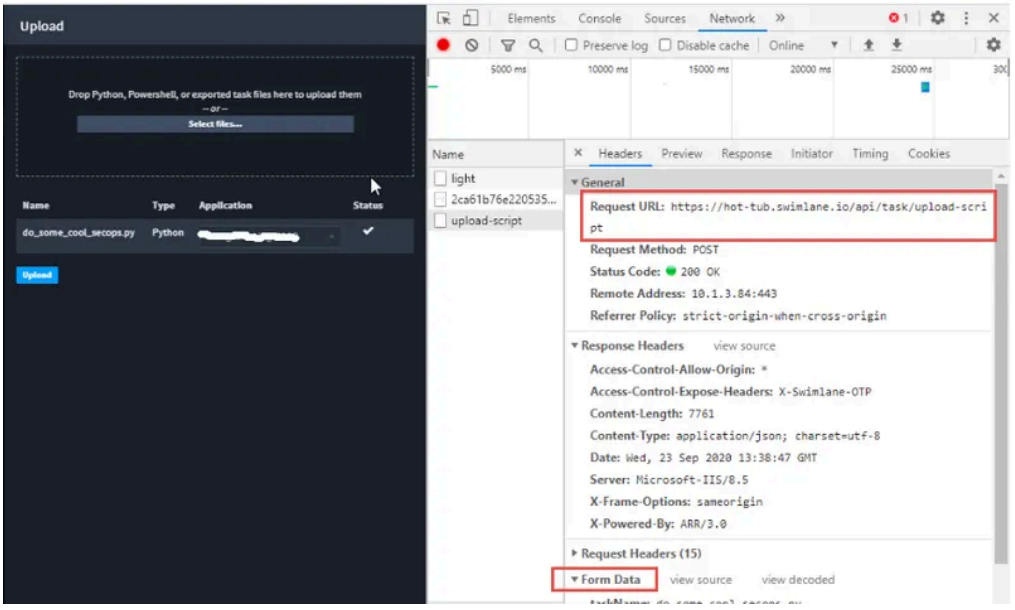
19.1. How to upload a Python script via the POST /task/upload-script endpoint

Many of the Swimlane RESTful API endpoints require JSON payloads. However, the POST /task/upload-script endpoint requires multipart/form-data.

To learn how to interact with this endpoint, or any other, it's best to inspect the Swimlane UI's manner of interacting with the endpoint:



Here is the successful POST:



19.11. How to update a Selection field definition via the RESTful API

When altering a Swimlane application via the RESTful, use the PUT /app/id endpoint. A common pitfall, when creating/altering Selection fields within an app via code, is that the burden is on the client code to make sure that every value defined for a Selection field must have a unique identifier. The Swimlane user guide doesn't specify that such identifiers must be generated, and it does not give guidance on their proper form.

The code for generating the identifier can be found here. This code will be included in the following sample illustrating how to fetch an existing application definition and alter an existing Selection field. The field's original values are "red", "yellow", and "blue". The snippet below removes the value "blue" and adds two others, "green" and "orange":

```
from swimlane import Swimlane
import json
import copy
import time
import binascii
import os


def random_objectid():
  """Returns a randomly generated MongoDB ObjectId."""
  timestamp = '{0:x}'.format(int(time.time()))
  rest = binascii.b2a_hex(os.urandom(8)).decode('ascii')
  return timestamp + rest


# Fetch the application's definition via the Swimlane Driver:

swimlane = Swimlane('https://HOSTNAME', 'USER', 'PASSWORD', verify_ssl=False)
app = swimlane.apps.get(name='APP-NAME')

# Fetch the Selection field's definition and print its values, "red", "yellow",
and "blue":

selection_field_def = app.get_field_definition_by_name('Selection')
#print(json.dumps(selection_field_def, sort_keys=True, indent=4, separators=(",",
": ")))


# Find and remove "blue", and print to confirm success:

for field_def in app._raw['fields']:
    if field_def['id'] == selection_field_def['id']:
        count = 0
        for val in field_def['values']:
            if val['name'] == 'blue':
                del field_def['values'][count]
                break
            count += 1
print('Blue has been removed\n\n')
print(json.dumps(app._raw['fields'], sort_keys=True, indent=4, separators=(",", ":
")))

# Add two new values, and print to confirm success:
```

19.21. SSDEEP tasks hanging and generating a large Hangfire queue

SSDEEP tasks are taking longer to process (up to over 24 hours).
The following errors are seen from the Hangfire page:

```
ProxyError: HTTPSConnectionPool(host='swimlane.corpzone.internalzone.com',
port=443): Max retries exceeded with url: /api/user/authorize (Caused by
ProxyError('Cannot connect to proxy.',
NewConnectionError('<urllib3.connection.HTTPSConnection object at
0x0000000003232C88>: Failed to establish a new connection: [Errno 11001]
getaddrinfo failed
```

- Dropping the Hangfire JobGraph collection db.hangfire.jobGraph.drop() does not resolve the issue.
- Restarting the Swimlane task service on Windows seems to help resolve the issue for just a few minutes, then the issue starts again.
- Restarting the Tasks server on a Linux system with docker stop sw_tasks and docker start sw_tasks also helps for a few minutes, then the issue starts again.

The issue lies in the SSDEEP integration task python script.

Workaround:

Modify the following in the python integration script:

1. The Loopback time:

The loopback_time is used to create a record filter. Then a “for loop” will run through the filter and process the data. If there are thousands of records that come up in the search filter, this may cause the task to be stuck in a loop for hours, causing a huge queue.

The loopback_time variable is set to 24 hours (1 day) by default . Set this to fewer hours between 1 and 12 hours.

2. Comment out the lines for Deduped Email and Tracking Id lines in the fuzzymatch():

At times, depending on what you get from the logs, you may need to comment out the lines for NOT_EQ for Deduped Email and Tracking Id:

```
records.filter('Deduped Email', NOT_EQ, 'Yes')
records.filter('Tracking Id', NOT_EQ, trackingId)
```

Mongo profiler logs indicated that using the NOT_EQ boolean was causing the indexing to fail and was slowing mongo lookup in the records in Hangfire, causing a huge processing queue.
Comment the lines out and replace both lines with the “for loop” statements below:

```
for record in records:
    if record.tacking_Id == trackingId:
        continue
    if record['deduped Alert'] == 'YES':
        continue
```

19.31. Introduction to Swimlane Programming

Background

Building security solutions in Swimlane consists of:

1. Defining applications to store your data and present it to users
2. Optionally
    1. Defining the application's workflow to both:
        1. Make each application record more interactive
        2. Automate tasks revolving around the record (enriching the record, sending external notifications, etc.)
    2. Adding Swimlane-provided Plugins (see here and here) to enrich records or integrate with external systems
    3. Writing custom scripts to enrich and integrate in novel ways

This article offers an introduction to custom scripting with Swimlane.

Internal Scripts

Two types of scripts can be authored to execute inside of Swimlane: Python and Powershell.

Like Swimlane Plugins, these scripts can be invoked by:

- Task triggers of various sorts
- Workflow actions of type Trigger Integration

They can interact with the Swimlane Platform via the following:

- TASK API
    - See Swimlane Documentation's TASK API resource
    - See the Support Portal's TASK API article
- REST API
    - See Swimlane Documentation's REST API resource
    - See the Support Portal's REST API and Driver article

Powershell scripts are limited because the TASK API doesn't support them as fully as it does for Python, and because they must interact with the RESTful API without the aid of any Swimlane-provided library.

On the other hand, Python scripts have more robust support for the TASK API, and they can leverage the Swimlane Driver (a Python library that makes it easier to write Swimlane RESTful API client code).

19.41. Gotenberg Chromium conversion failed

Issue Identified: A code change in Swimlane 10.20.1 prevents the download or distribution of any dashboard or report. The following errors may be observed:

```
Gotenberg Chromium causing the error "Failed to export the report pdf. Gotenberg Chromium conversion failed conversion failed" when downloading reports
```

or

```
Unhandled exception occurred while processing API request: Gotenberg Chromium conversion failed.
```

As an On-prem customer you may use this command below to patch the gotenberg service until the 10.20.2 hotfix release is made available and deployed in your environment.

Important Notes:

- As an Embedded cluster customer, you have gotenberg running in the default namespace
- As an Existing cluster customer, you will most likely have Swimlane deployed in a custom namespace so update the namespace in the kubectl patch command...the argument after the -n switch.

```
kubectl -n <namespace> patch deployment gotenberg --type=json -p='[{"op": "remove","path": "/spec/template/spec/containers/0/securityContext/readOnlyRootFilesystem"}]'
```

This change will persist until the next version (10.20.2) is deployed from the KOTS Admin UI.

19.42. MongoDB query/script for Swimlane to parse all records for an application and automatically remove inaccessible attachments (10.x)

Below is the query to clean up records with corrupted or inaccessible attachments by validating FileIds against the GridFS (fs.files) collection.

The query will:

1. Remove invalid attachments from both Values and ValuesDocument.
2. Retain valid attachments in the respective fields.
3. Ensure no valid data is inadvertently removed.

```
const BATCH_SIZE = 1000; // Define batch size for updates
let totalProcessed = 0; // Track total processed records

// Fetch all Attachment Fields
```