UNKNOWN

CYBER

ThreatConnect User Guide

Revised: 1/2/25

1. Introduction

This Unknown Cyber app integrates with the ThreatConnect platform to analyze binary files, retrieve file information, and generate Yara rules. It allows users to upload binaries, retrieve detailed results, and perform analysis within ThreatConnect Playbooks. This integration supports threat intelligence, malware analysis, and yara-based detection in workflows.

2. Installation

2.1 Requirements

These requirements are required to use **Unknown Cyber** App in ThreatConnect Playbooks:

- Access to a ThreatConnect instance
- Access to ThreatConnect Playbooks
- A Unknown Cyber API key
- **Unknown Cyber** App installed in your ThreatConnect instance

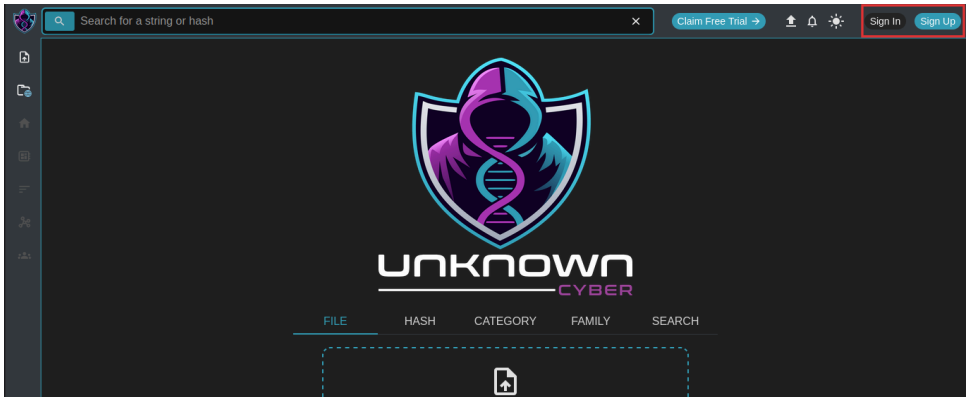
2.2 App Installation

The **Unknown Cyber** App is available on ThreatConnects GitHub. Download the .tcx file and install it following the ThreatConnect System Administration Guide.

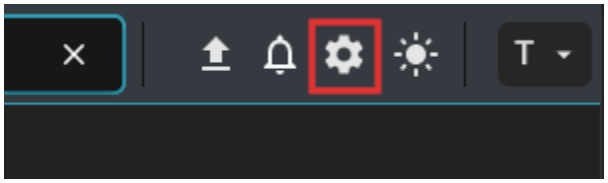
2.3 API Key Setup

To use the **Unknown Cyber** App, you need an Unknown Cyber API Key. It is then recommended to add your **Unknown Cyber API Key** to ThreatConnect's Key Vault.

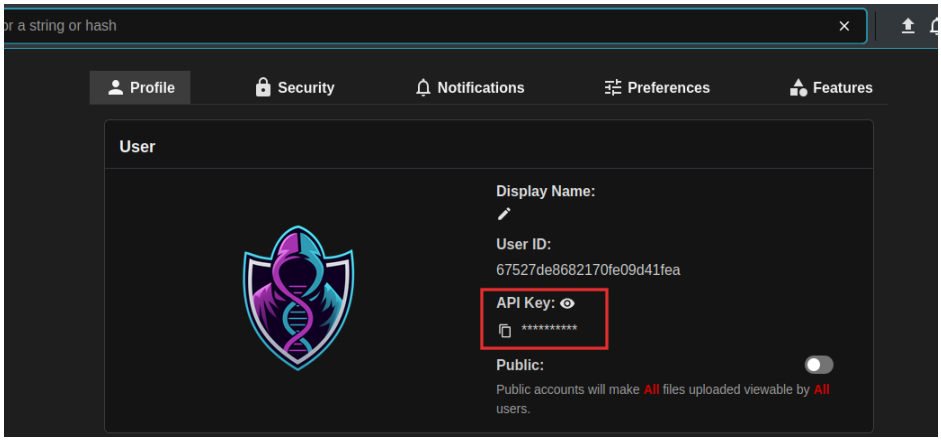
1. Go to www.unknowncyber.com and sign in or sign up



2. Click the **Settings Cog** in the upper right hand corner.



3. Copy your **API Key**.



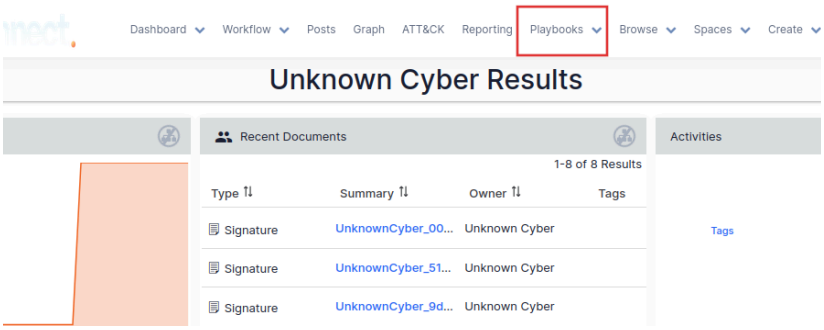
4. Go to your ThreatConnect instance and add the API Key as a Variable in either the “My Profile” or “Org Settings” section. The instructions for this can be found in ThreatConnect’s documentation of “My Profile: Variables Tab”.

3. Configuration

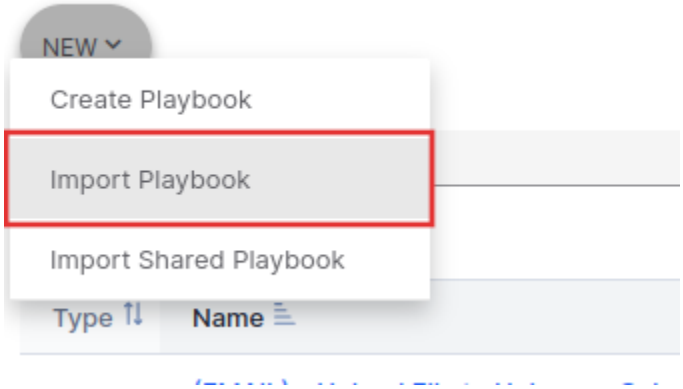
When configuring a playbook, you can build your own, or use one of ours as a starting point. Below, we show you how to set up both.

3.1 Importing Playbook

1. Click on Playbooks in the top menu.



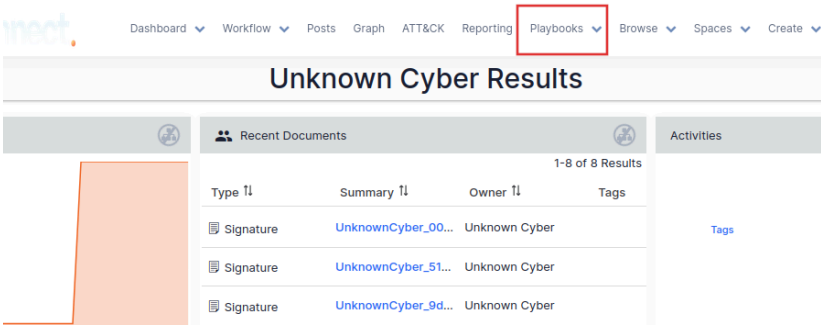
2. Then hover over the **New** button on the left side and click **Import Playbook**.



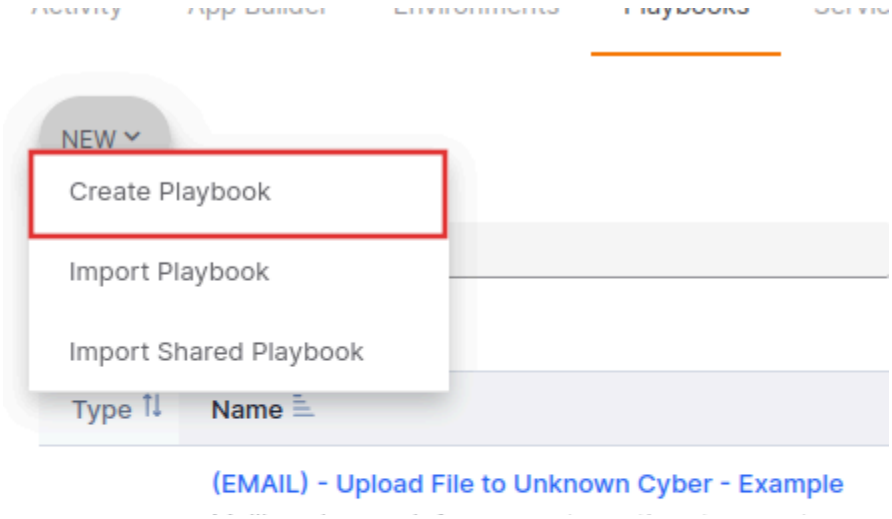
3. Select the **.pbxz** playbook file you want to import and follow the onscreen instructions ThreatConnect provides.

3.2 Building a Playbook

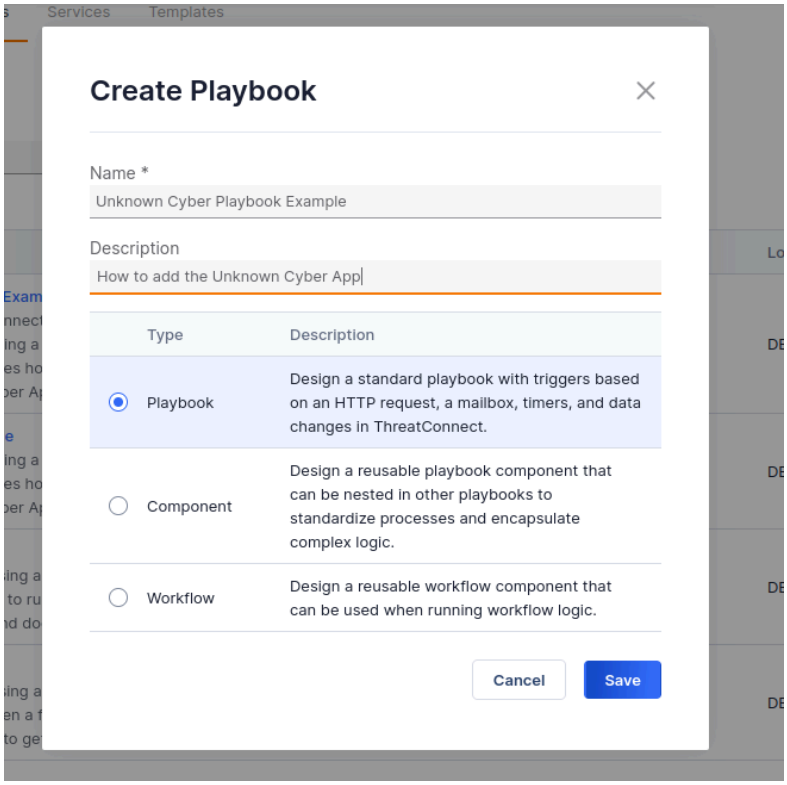
1. Click on Playbooks in the top menu.



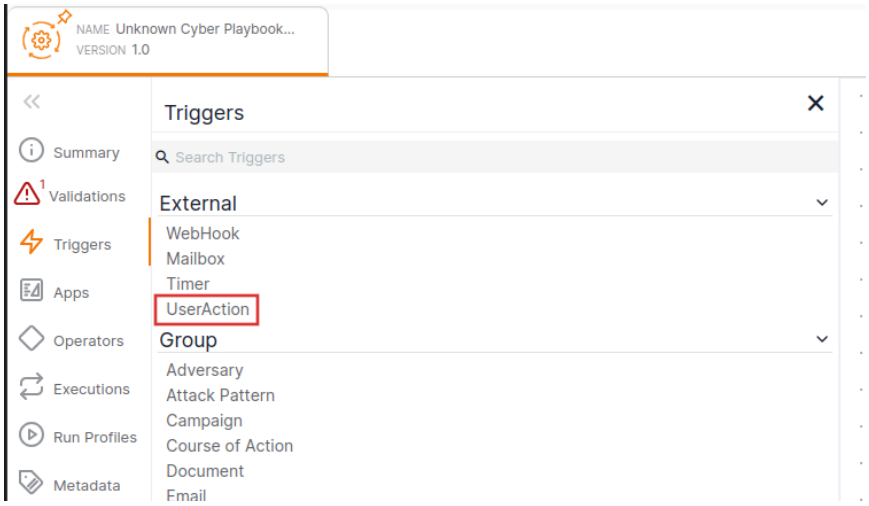
2. Then hover over the **New** button on the left side and click **Create Playbook**.



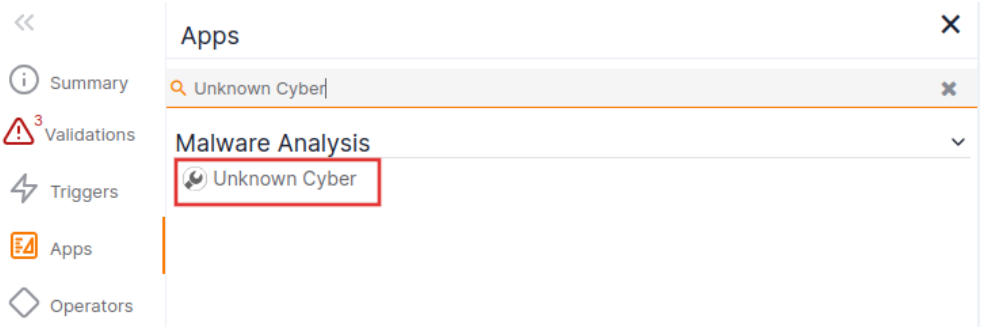
3. Enter a **Name** and **Description** for the playbook and select the **Playbook Type**. Then click **Save**.



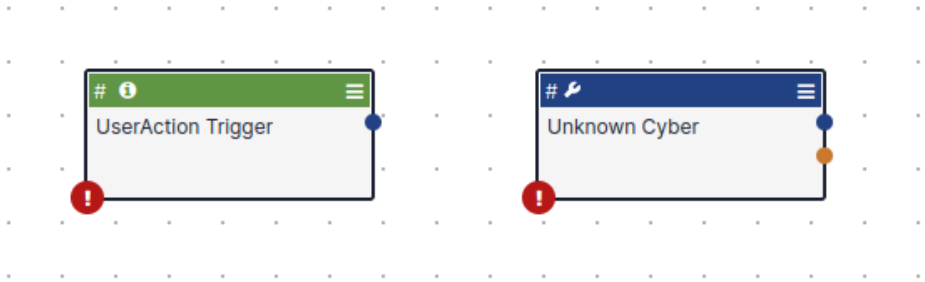
4. Double click on the **UserAction** Trigger. This will add the trigger to the playbook and let you run the playbook later.



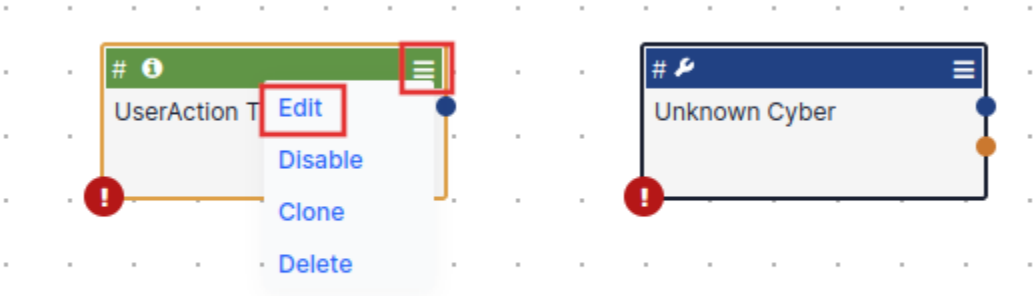
5. Select **Apps** from the left sidebar, and then enter **Unknown Cyber** into the Search Bar. After, double click on **Unknown Cyber** to add it to the playbook.



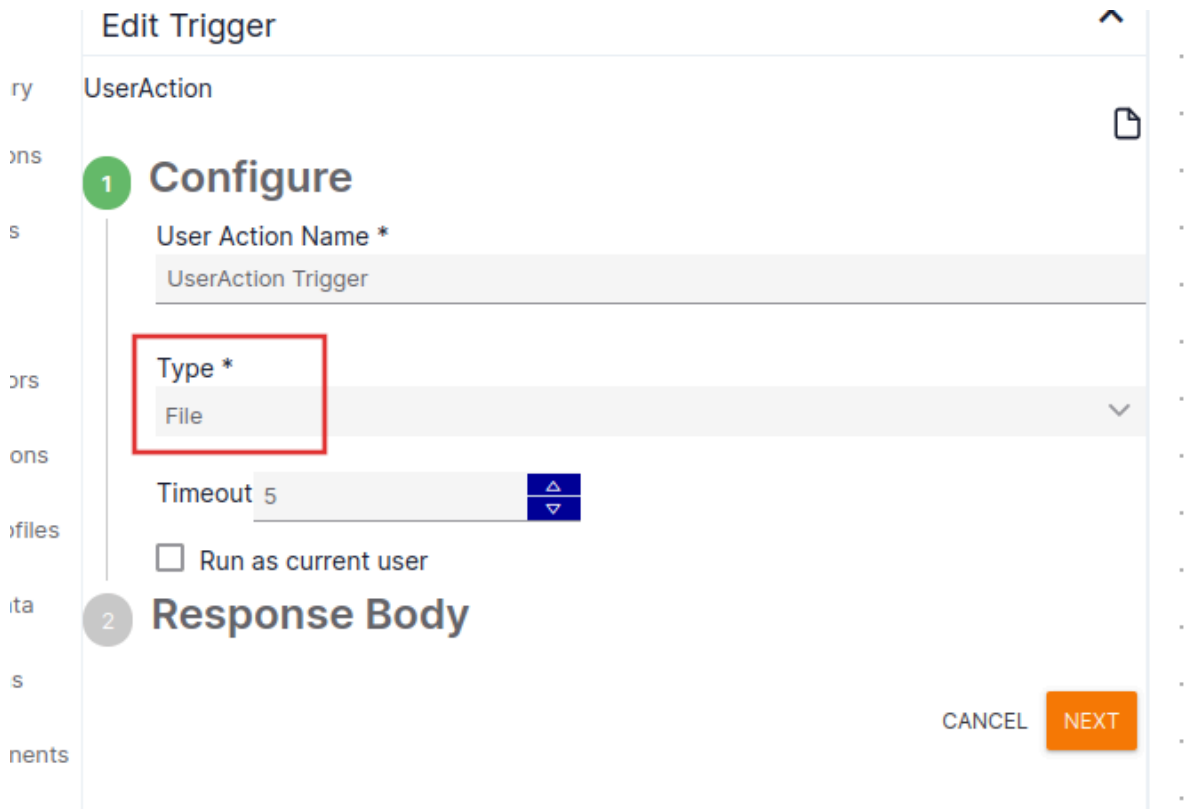
6. Your playbook should look like this with the trigger and Unknown Cyber app.



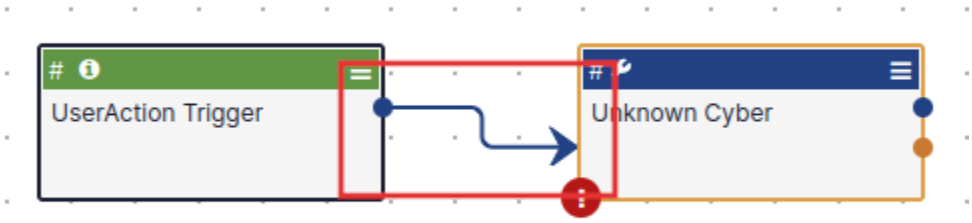
7. Now click on the **three bars Icon** in the upper right side of the trigger. Then click **edit**.



8. A **Edit Trigger** window should appear as shown below. In this window, click on **Type** and select **File**. After that, click **Next**, and then **Save** to save the changes.

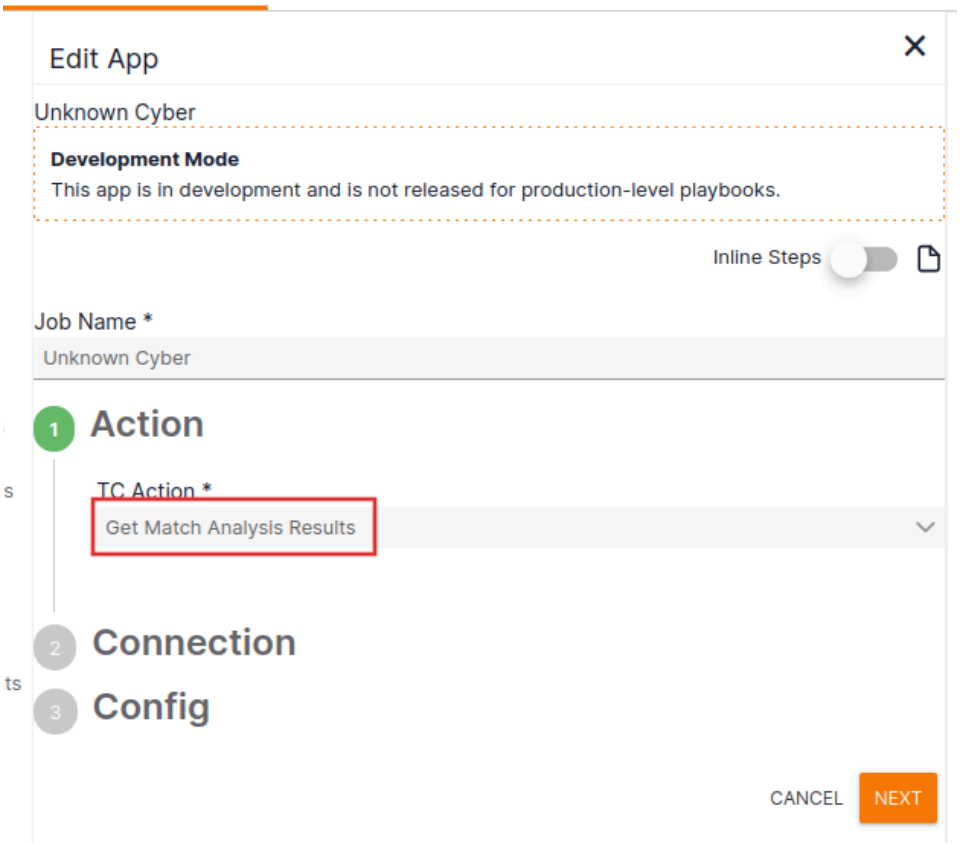


9. Now connect the **UserAction** Trigger to **Unknown Cyber**. Do this by clicking on the blue dot of the Trigger and dragging your mouse to Unknown Cyber.

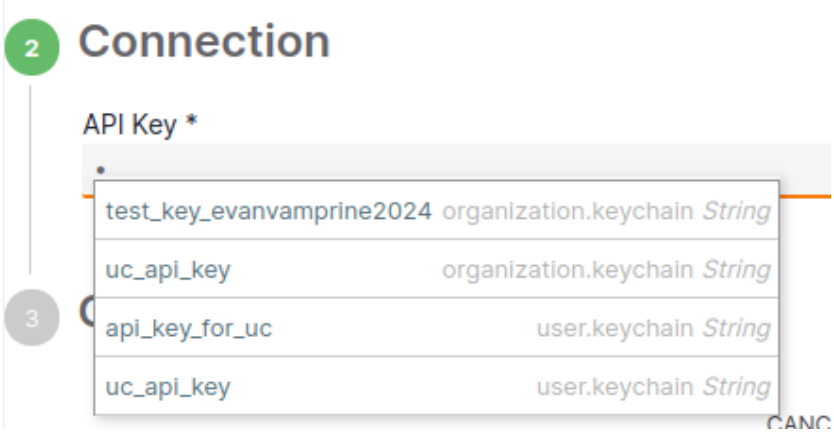


10. Now click on the **Three Bars Icon** of **Unknown Cyber** and then click **edit**.

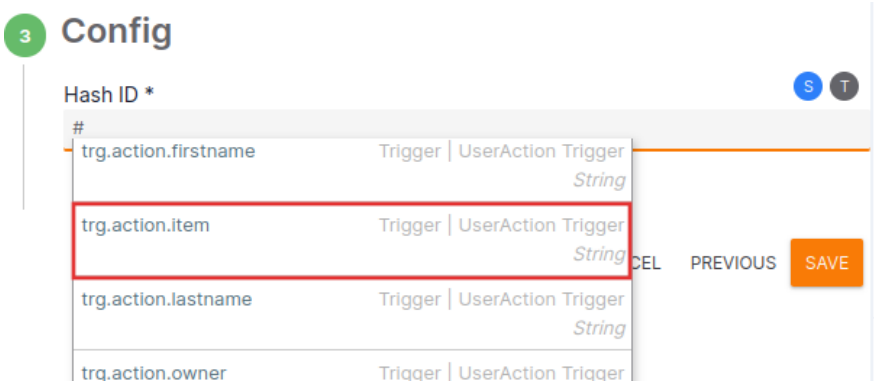
11. When the Edit App window opens, you will see three steps: Action, Connection, and Config. These steps will determine which App Action, API Key, and Configuration you use for this App. We will use **Get Match Analysis Results** as an example. Select if from the **TC Action**. Then click **Next**.



12. This is where you will enter your **Unknown Cyber API Key**. If you added your API Key as a ThreatConnect Variable, then type **"$"** and you should see a list of your keys. If not, you can directly type in your API Key. Then, click **Next**.

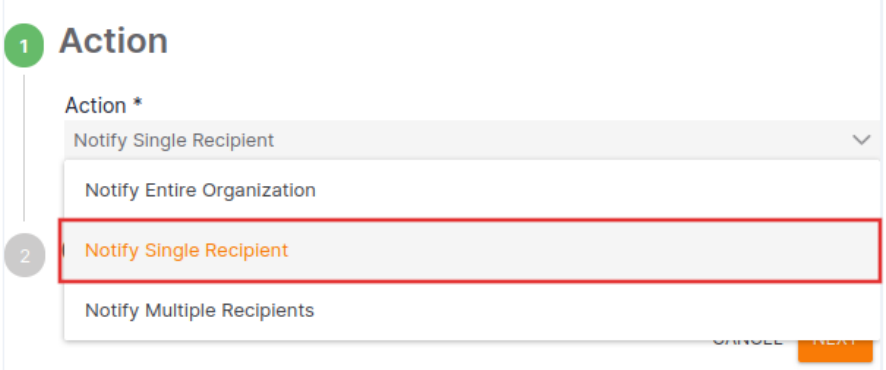


13. In the **Hash ID** section, type **"#"** to show a list of variables you can select. We will select the **trg.action.item** for this example. Then, click **Save**.

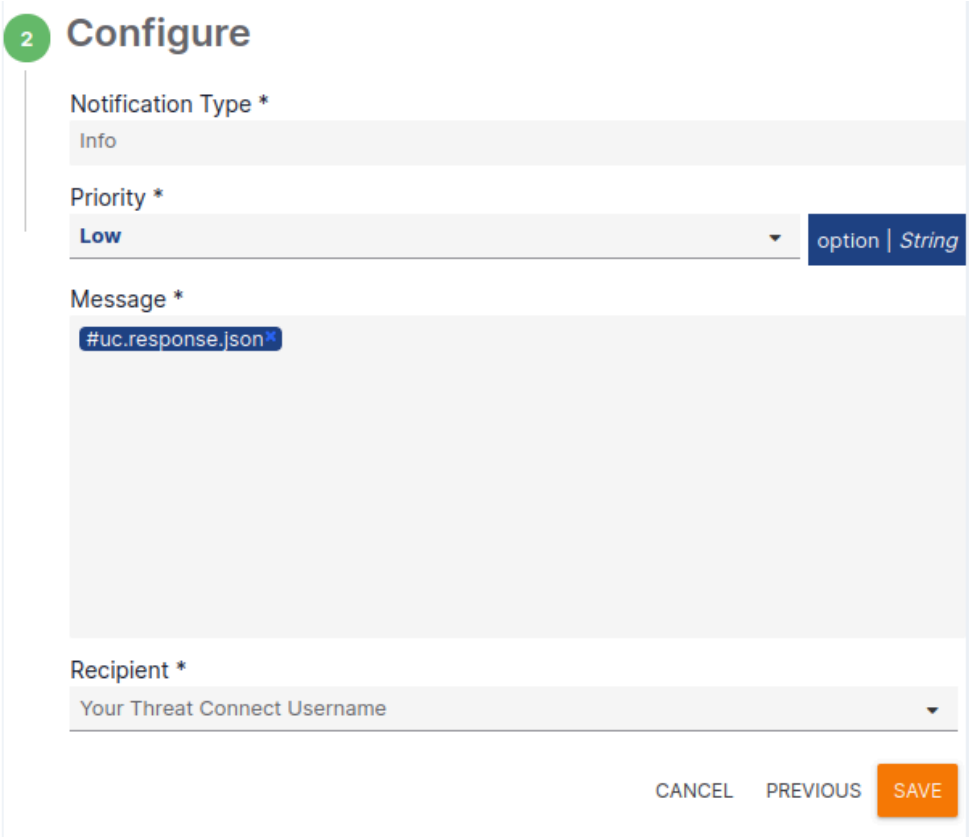


14. To see the results quickly, we will add the **ThreatConnect Notification** app behind Unknown Cyber. Follow step 5 replacing **Unknown Cyber** with **ThreatConnect Notification**.

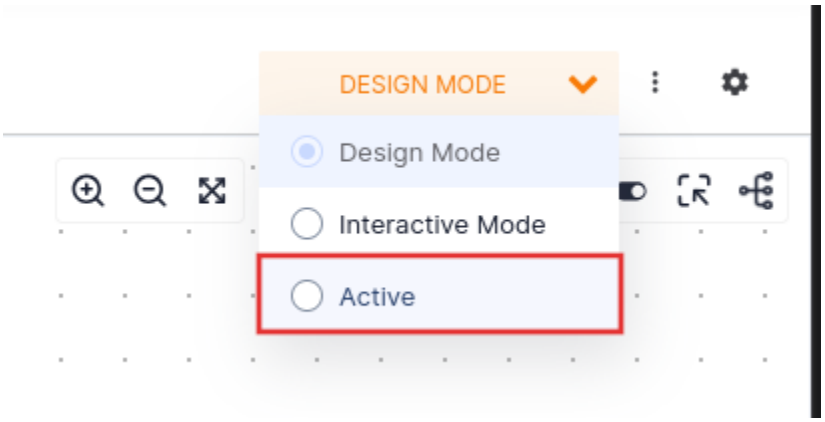
15. Now, **double click** the ThreatConnect Notification and select **Notify Single Recipient** for this example. Then, click **Next**.



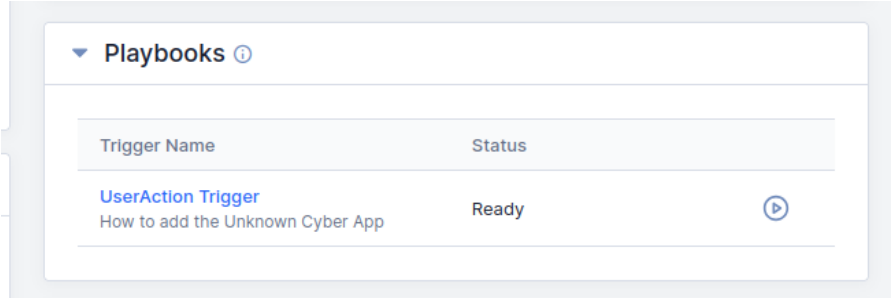
16. Fill in the four sections. Then, click **Save**.
    a. **Notification Type**: “Info”
    b. **Priority**: “Low”
    c. **Message**: “#uc.response.json”
    d. **Recipient**: Your ThreatConnect Username

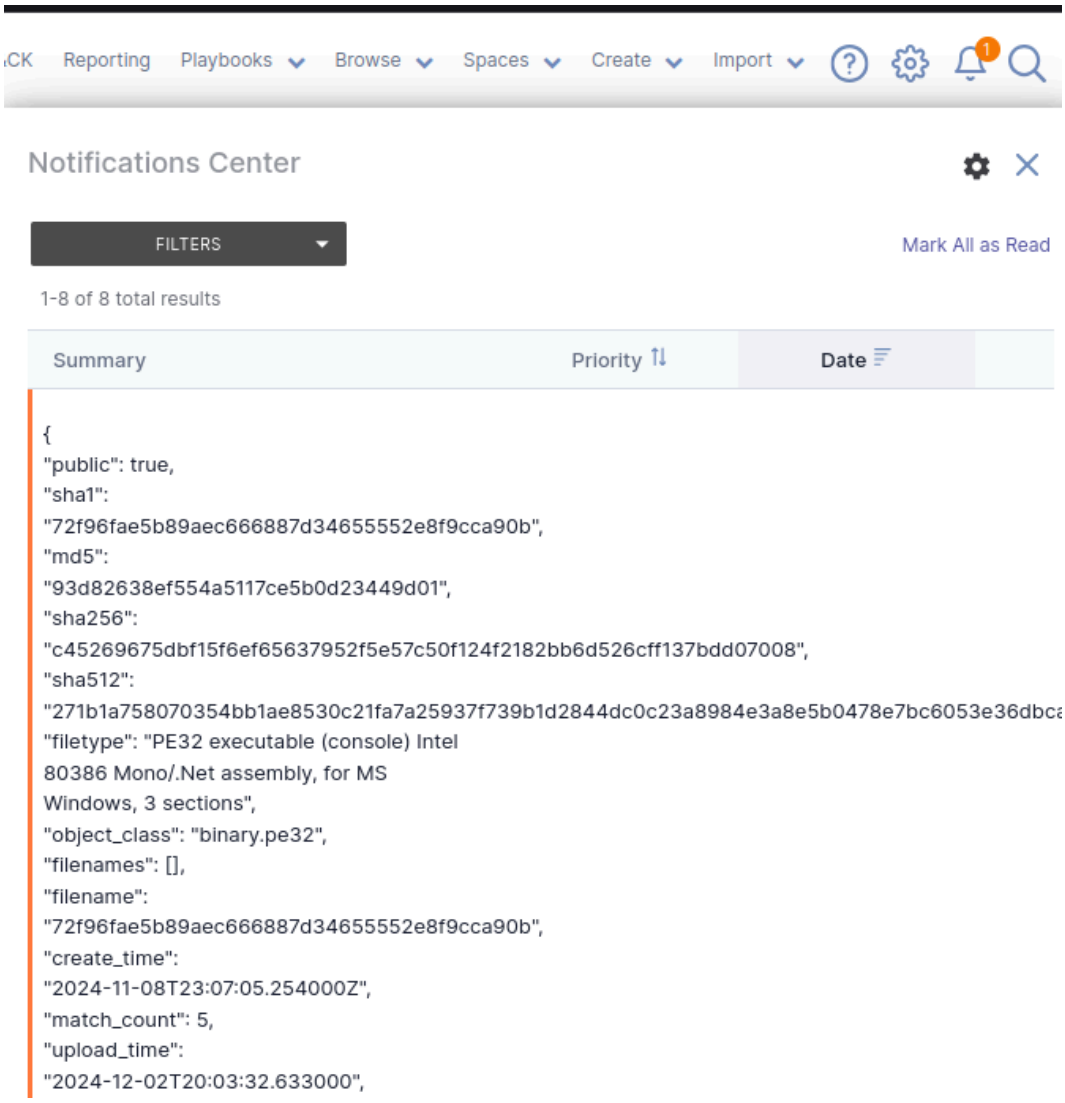


17. Lastly, set the playbook to **active** in the upper right corner.



18. You can test this playbook by going to **Browse**, **Indicators: File**, and then selecting a **File**. Once on a File's page, the **Playbooks** section should contain a **UserAction Trigger** and the **Name of the Playbook** created. Click the **play button** and in a few seconds, a notification will be available in the top right of the ThreatConnect page with the response. You might need to click the **bell icon** to make the notification appear.





4. App Actions

Actions are optional paths used in a playbook. They are selected after the Unknown Cyber app is loaded into the playbook. Unknown Cyber currently offers four actions and only one action can be selected at a time.

Note: The two tables below show default Inputs and Outputs that are available on all Unknown Cyber app instances.

Input	TC Type	Description
Job Name *	String	This is the name given to this particular instance of a job
API Key *	String, $Keychain	A user's Unknowncyber API Key
TC Action *	Dropdown	This is the action to be performed.

Output	TC Type	Description
tc.action	String	The type of action chosen for this job
uc.response.status_code	String	The api request's status code.
uc.response.success	String	True if the api request was successful and false if not.
uc.response.errors	String	The api request's error message if the request fails.
uc.response.json	String	The raw JSON response from Unknown Cyber's api.

4.1 Get Match Analysis Results

Retrieve Unknown Cyber's analysis for a specified file hash. For a more granular look at a hash's data, use the uc.response.json.

Input	TC Type	Description
Hash ID *	String, $Text	The hash of a file to receive details on. Must be a MD5, SHA1, SHA256, or SHA512.

Output	TC Type	Description
uc.response.md5	String	The md5 of the requested hash
uc.response.sha1	String	The sha1 of the requested hash.
uc.response.sha256	String	The sha256 of the requested hash.
uc.response.sha512	String	The sha512 of the requested hash.
uc.response.threat_level	String	The threat level of the hash. **Possibilities**: *New, Caution, Low, Medium, High*
uc.response.evasiveness	String	A value from 0 to 1.0 indicating how many scanners have seen the hash.
uc.response.category	String	Returns the top category for the hash.
uc.response.family	String	Returns the top family for the hash.
uc.response.self_link	String	The link to the resource using Unknown Cyber's Api.
uc.response.match_count	String	The number of matches for the requested hash.
uc.response.children	StringArray	A list of children. **Example:** An archive's contents.

4.2 Get Processing Status

Retrieves the processing status for a hash. Once the file is done processing, it will set the uc.response.processing_completed to *True*. For a more granular look at the current file status, use the uc.response.json.

Input	TC Type	Description
Hash ID *	String, $Text	The hash of a file to receive details on. Must be a MD5, SHA1, SHA256, or SHA512.

Output	TC Type	Description
uc.response.processing_completed	String	Has the processing of the file through Unknown Cyber's system finished. **Possibilities:** *True, False*

4.3 Create Byte Code Yara

Automatically generates a Yara rule for the specified hash.

Input	TC Type	Description
Hash ID *	String, $Text	The hash of a file to receive details on. Must be a MD5, SHA1, SHA256, or SHA512.

Output	TC Type	Description
uc.create_yara.yara_rule	String	The automatically created yara rule for the hash.
uc.create_yara.yara_name	String	The name for the automatically created yara rule.

4.4 Get Matched Malicious Hashes

Gets a list of hashes that match the requested hash. By default, only perfect matches, 1.0, are retrieved. This can be adjusted using the Max and Min Similarity options.

Input	TC Type	Description
Hash ID *	String, $Text	The hash of a file to receive details on. Must be a MD5, SHA1, SHA256, or SHA512.
Min Similarity *	String, $Text	Minimum similarity threshold between 0.7 and 1.
Max Similarity *	String, $Text	Maximum similarity threshold between 0.7 and 1.
Response Hash	Dropdown	Allows other hashes to be returned. **Default:** *SHA1* **Possibilities:** *MD5, SHA1, SHA256, SHA512*
No Match Error	Boolean	Setting this to True will cause the app to throw an error if there are no matches.

Note: If you enter a minimum similarity score higher than the maximum similarity score, the scores will be set to equal the Max Similarity entered.

Output	TC Type	Description
uc.response.match_list	StringArray	A list of matches between the Max Similarity and Min Similarity. If no matches are found, the value will be **None**.

4.5 Analyze Binary

Upload a binary sample to Unknown Cyber for analysis. Any children or siblings of an upload are also individually processed and the results will be returned when following the 3 playbook examples below.

Input	TC Type	Description
File Sample *	Binary, $File	Binary content to upload.
Filename	String, $Text	Name for the uploaded file.
File Password	String, $Text	Password for file extraction from encrypted archive.

Output	TC Type	Description
uc.response.md5	String	The md5 of the uploaded file.
uc.response.sha1	String	The sha1 of the uploaded file.
uc.response.sha256	String	The sha256 of the uploaded file.
uc.response.sha512	String	The sha512 of the uploaded file.

4.6 Get Bo LLM Behavior Report

Ask Unknown Cyber's AI assistant Bo for a human readable version of the byte code and give its analysis on the code.

Input	TC Type	Description
Hash ID *	String, $Text	The hash of a file to receive details on. Must be a MD5, SHA1, SHA256, or SHA512.

Output	TC Type	Description

Note: There are no custom output variables for Get Bo LLM Behavior Report. Use *uc.response.json*.

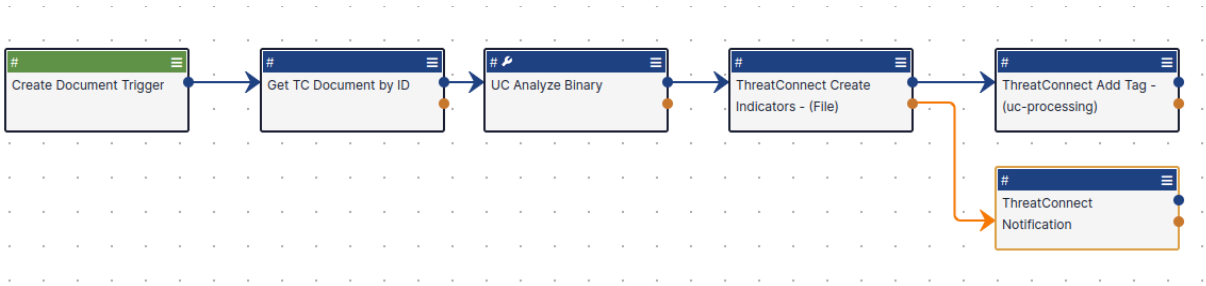
5. Playbook Examples

You can find example playbooks for Unknown Cyber on GitHub. These playbooks illustrate a basic automated workflow, including uploading a file to Unknown Cyber, monitoring its upload status, and generating automated Yara rules for matched files.

To install a playbook, navigate to the **Playbooks** tab in ThreatConnect. Click **New**, select **Import**, and choose the **.pbxz** playbook file you want to import. Follow the on-screen instructions to complete the playbook setup.

5.1 Upload File to Unknown Cyber

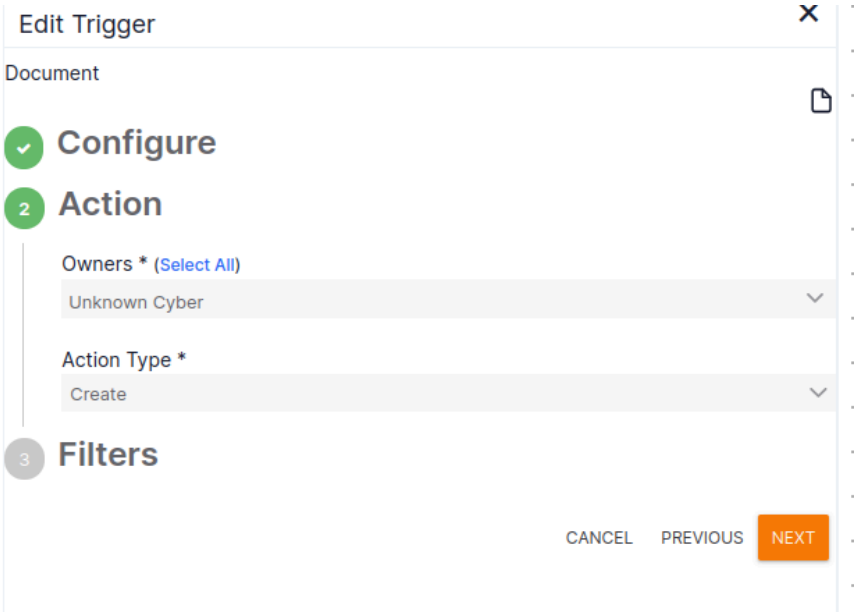
This playbook is the first out of three. This one handles taking an uploaded document from ThreatConnect and sending it to Unknown Cyber. Unknown Cyber will respond with basic analytics, and that data will be used to create a ThreatConnect File Indicator object. From there, the file indicator object will be tagged with “uc-processing”. If there is an error, such as the file already existing, then a notification will be sent in ThreatConnect indicating so.



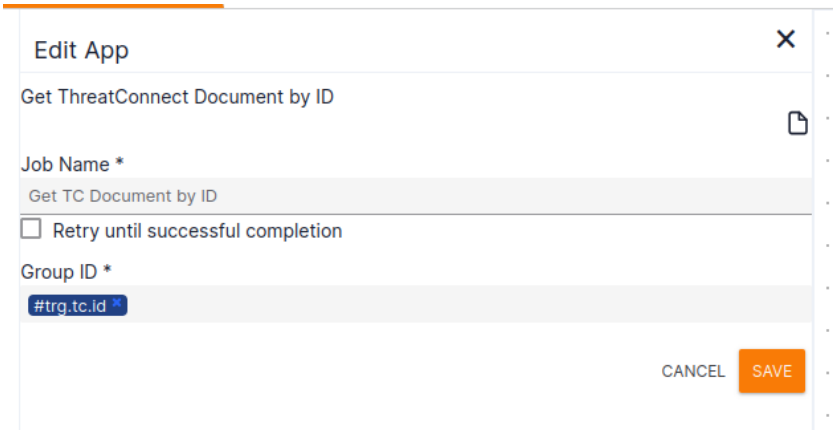
Steps

1. Use a **Document** Trigger and a **Get ThreatConnect Document by ID** App and connect them. Then configure them as shown below.

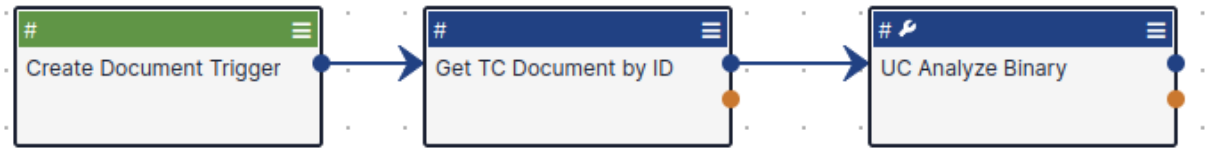
Create Document Trigger	
Owners	Which Intel Source you want to trigger the playbook.
Action Type	Create



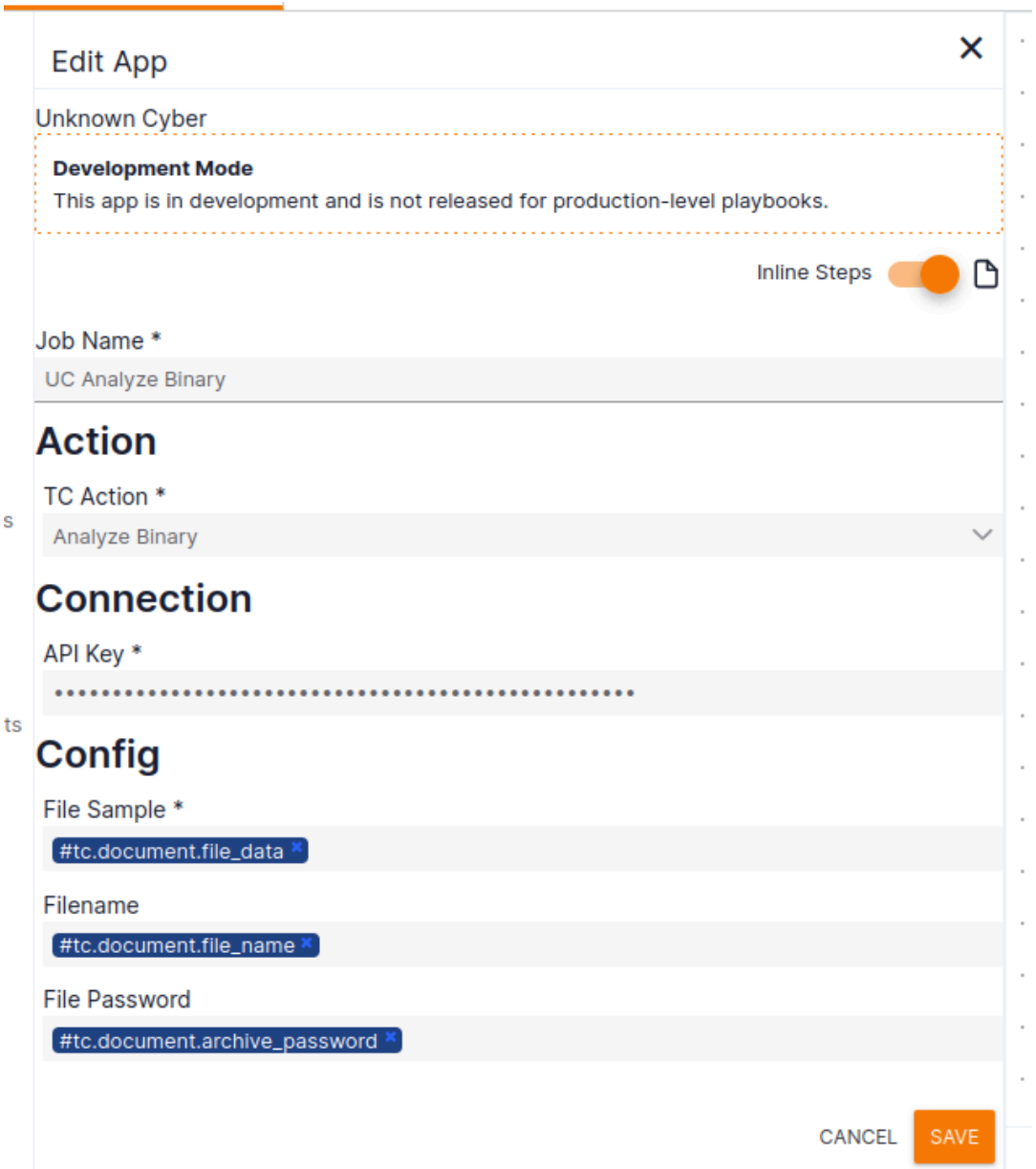
Get ThreatConnect Document by ID	
Group ID	#trg.tc.id



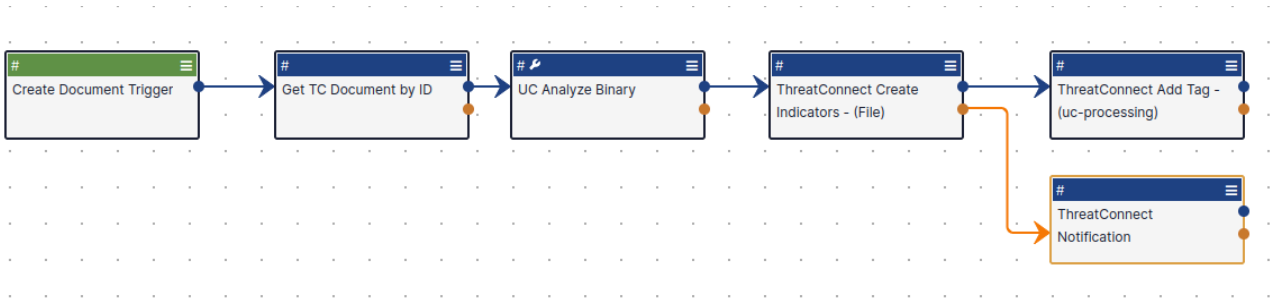
2. Add the **Unknown Cyber** app to the Playbook and connect it behind the **Get TC Document by ID**. Then use the configuration below to handle archive passwords and file names. This is where the document is Uploaded to Unknown Cyber to be analyzed.



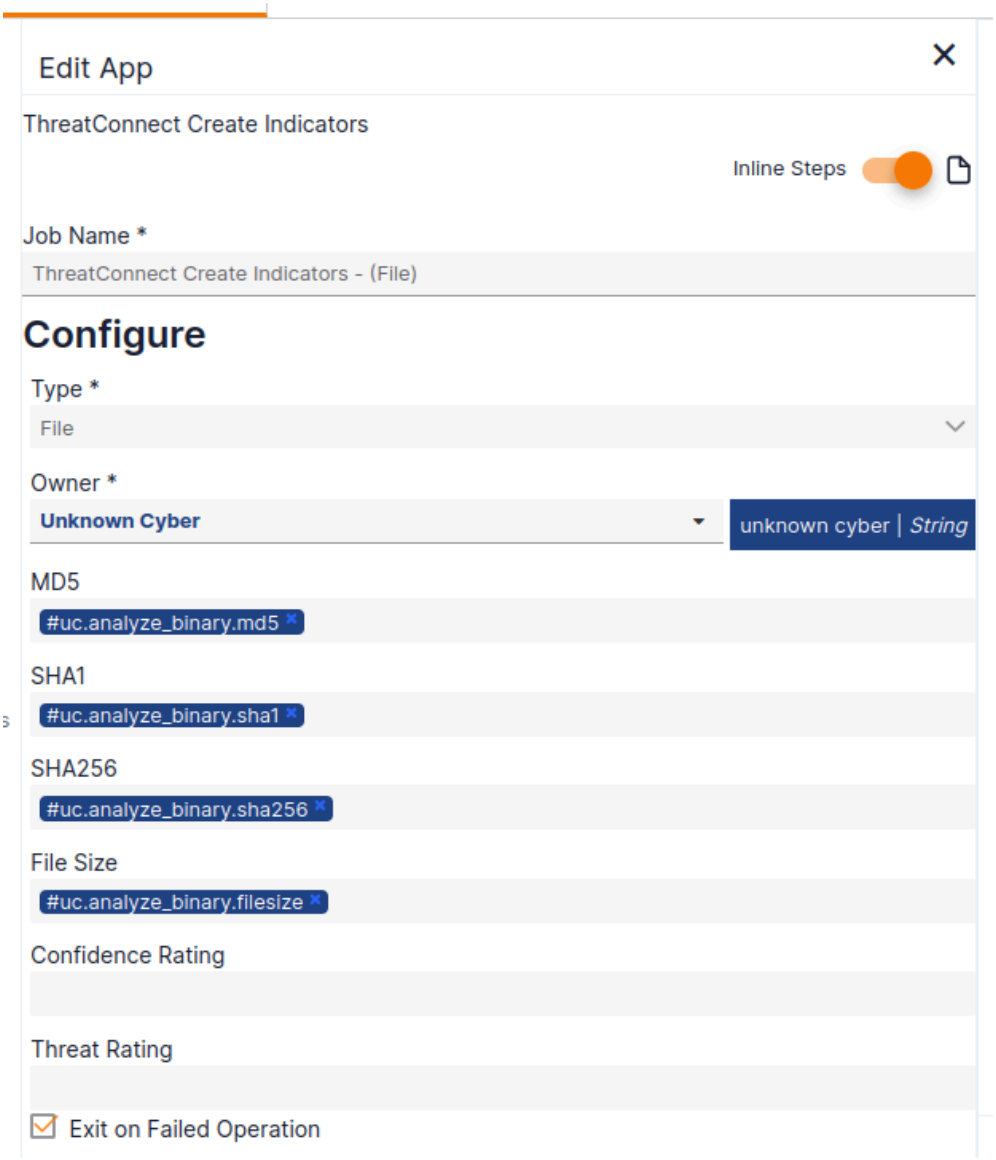
Unknown Cyber (UC Analyze Binary)	
TC Action	Analyze Binary
API Key	Your Unknown Cyber API Key
File Sample	#tc.document.file_date
Filename	#tc.document.file_name
File Password	#tc.document.archive_password



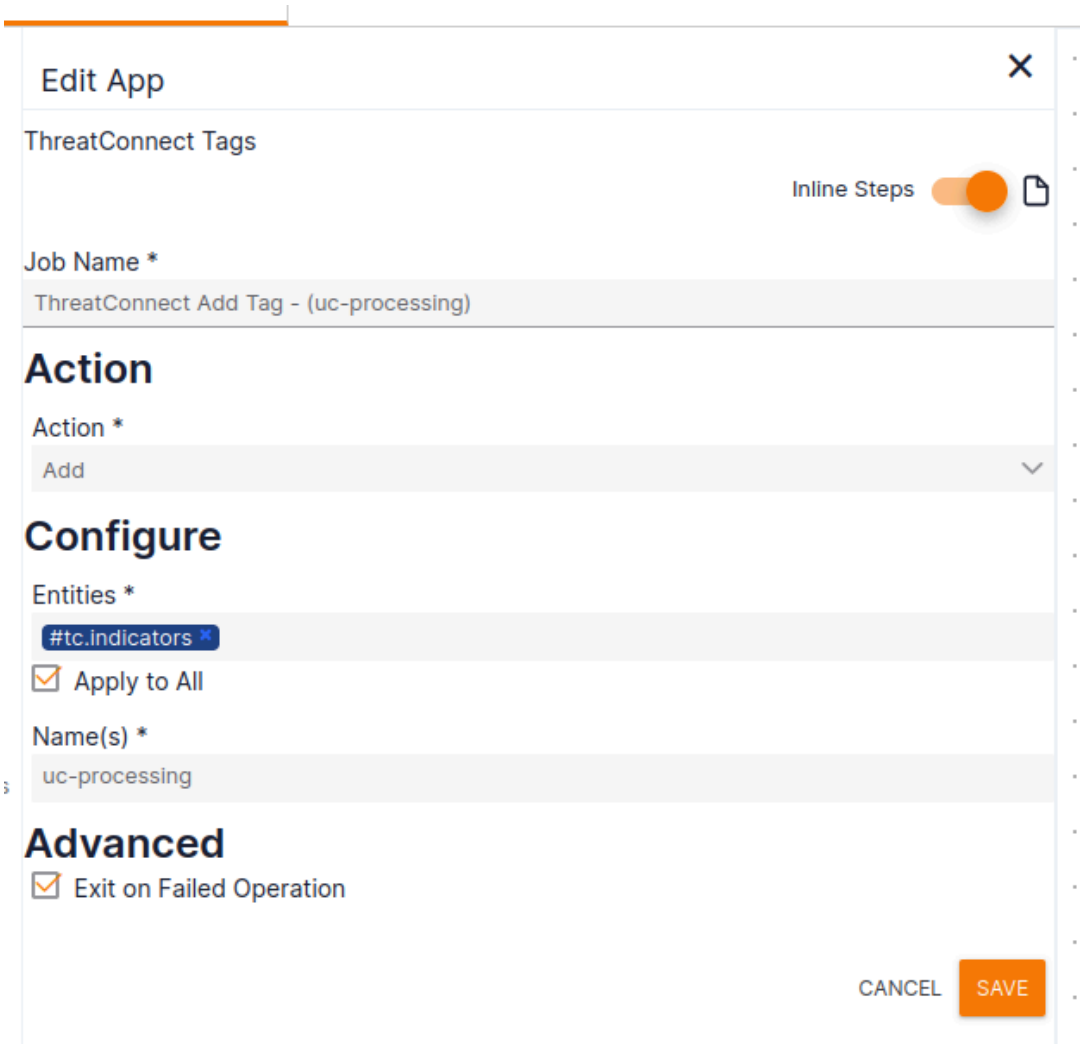
3. Lastly, add these three apps: **ThreatConnect Create Indicators**, **ThreatConnect Tags**, and **ThreatConnect Notification**. Connect and configure them as shown below.



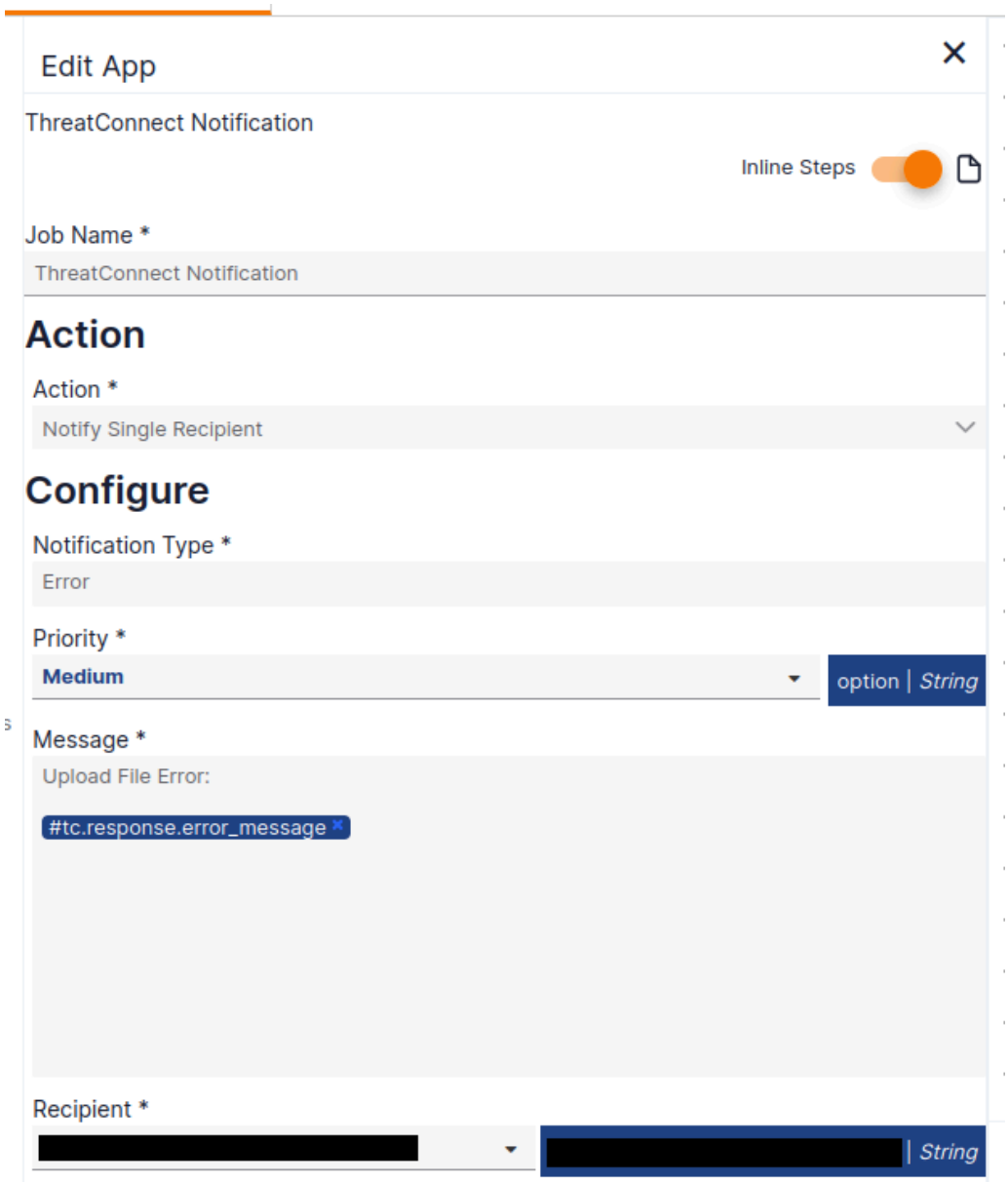
ThreatConnect Create Indicators	
Type	File
Owners	Which **intel source** should own this indicator.
MD5	#uc.analyze_binary.md5
SHA1	#uc.analyze_binary.sha1
SHA256	#uc.analyze_binary.sha256
File Size	#uc.analyze_binary.filesize
Exit on Failed Operation	True



ThreatConnect Tags	
Action	Add
Entities	#tc.indicators
Apply to All	True
Name(s)	uc-processing
Exit on Failed Operation	True

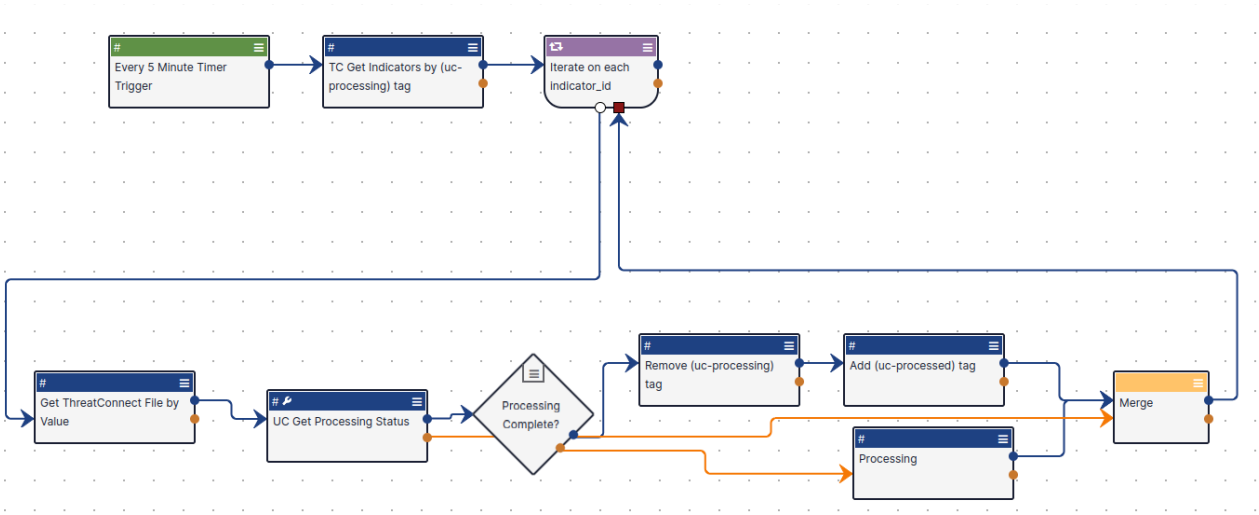


ThreatConnect Notification	
Action	Notify Single Recipient
Notification Type	Error
Priority	Medium
Message	Upload File Error: #tc.response.error_message
Recipient	Which user(s) should receive the Notification in your ThreatConnect Instance



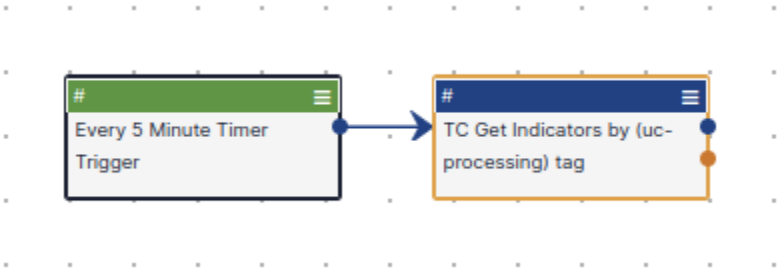
5.2 Check Processing Status

This playbook is the second out of three. This playbook runs on a 5 minute timer and loops through all the ThreatConnect File Indicators with a tag of *uc-processing*. For each File Indicator with the tag, a check processing status call is made to Unknown Cyber. If Unknown Cyber responds with *true*, indicating the file finished processing, then a tag of *uc-processed* will be added to the File Indicator and the *uc-processing* tag will be removed. If the File has not finished processing, the loop will simply continue and the playbook will retry the File Indicator in 5 minutes.



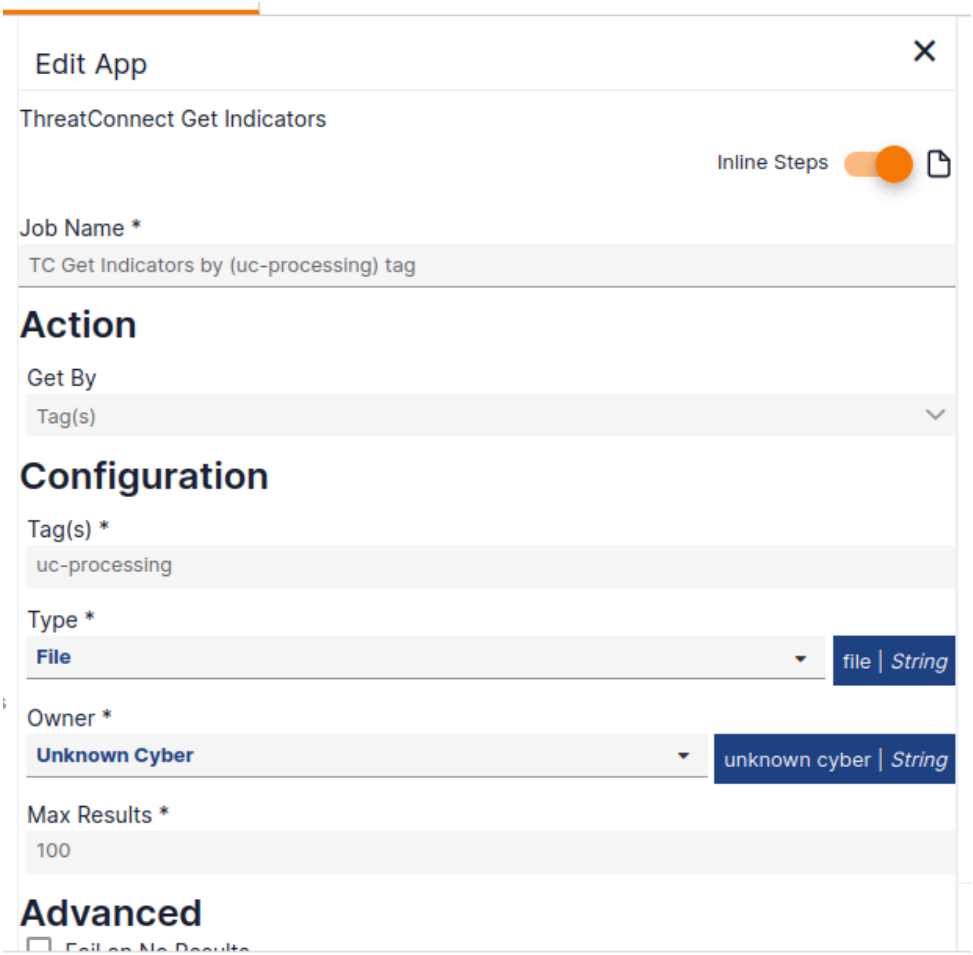
Steps

1. Use a **Timer** Trigger and **ThreatConnect Get Indicators** App and connect them. Then configure them as shown below.

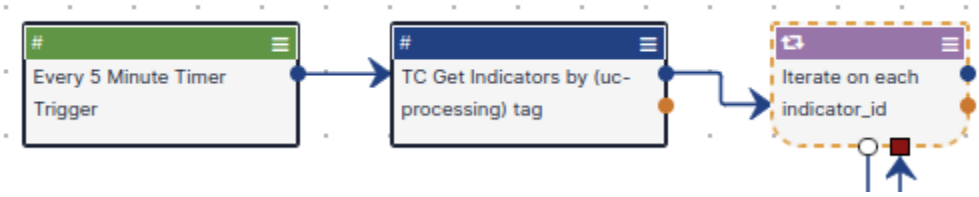


Timer	
Schedule	Advanced
Cron Expression	0 */5 * ? * *

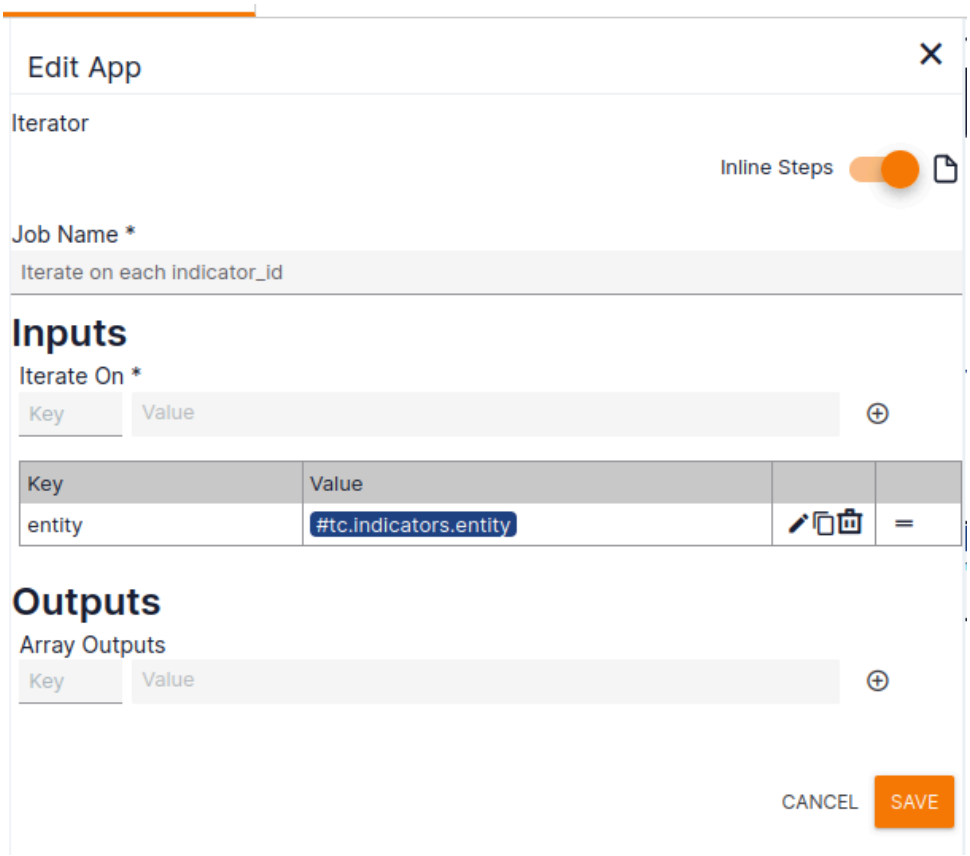
Timer	
Get By	Tag(s)
Type	File
Owners	Which **intel source** should own this indicator.
Max Results	100



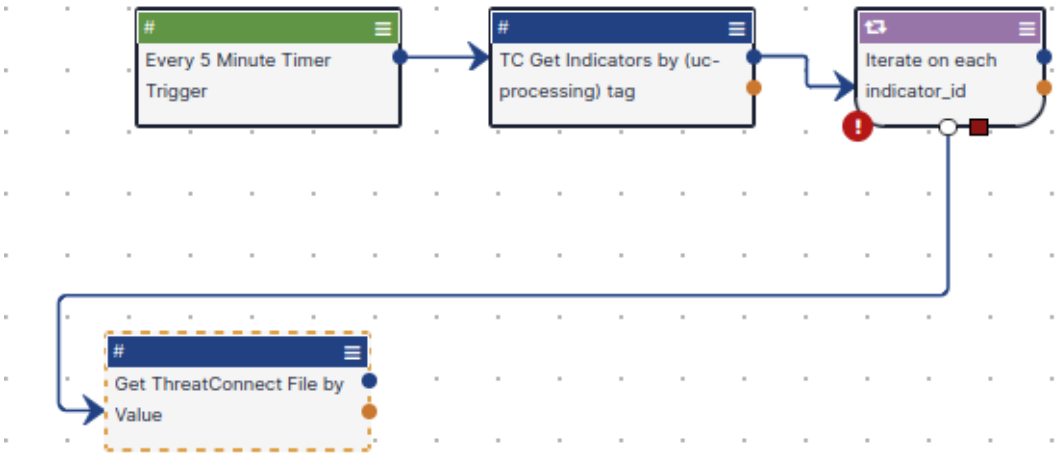
2. Now add an **Iterator** operator and connect it behind the **ThreatConnect Get Indicators**. Then configure it as shown below.



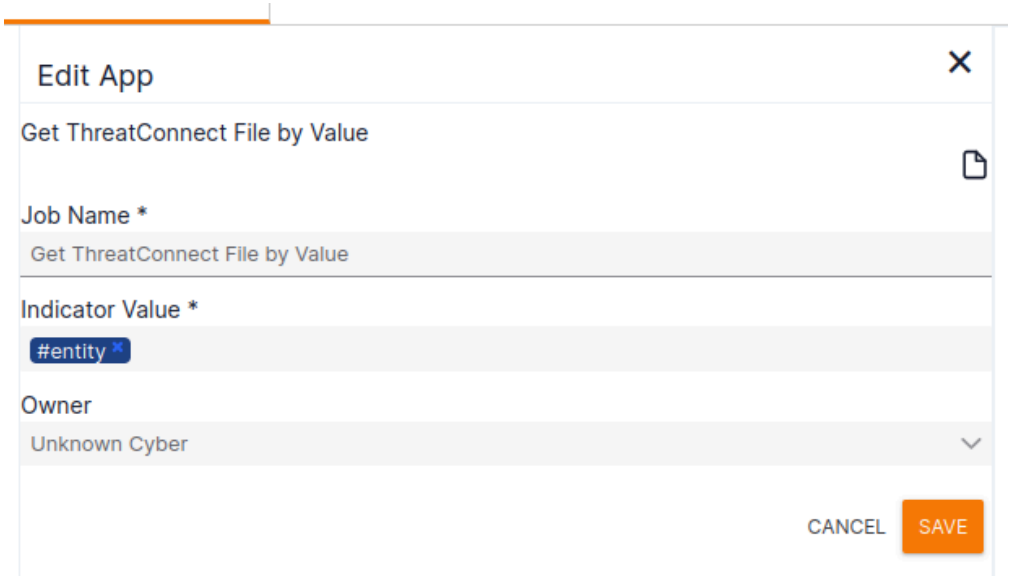
Iterator	
Inputs	
Key	entity
Value	#tc.indicators.entity



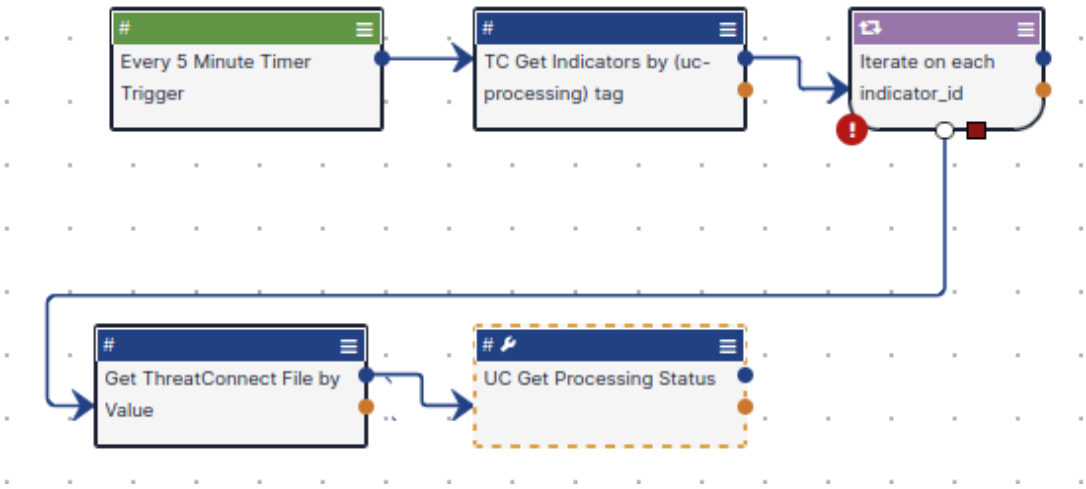
3. Next, add a **Get ThreatConnect File by Value** app and connect it to the start of the **iterator** loop. Then configure it as shown below.



Get ThreatConnect File by Value	
Indicator Value	#entity
Owner	Which **intel source** should own this indicator.



4. Now add the **Unknown Cyber** app and connect it after **Get ThreatConnect File by Value**. Then configure it as shown.



Unknown Cyber	
TC Action	Get Processing Status
API Key	Your Unknown Cyber API Key
Hash ID	#tc.file.value

Edit App

Unknown Cyber

Inline Steps

Job Name *

UC Get Processing Status

Action

TC Action *

Get Processing Status

Connection

API Key *

••

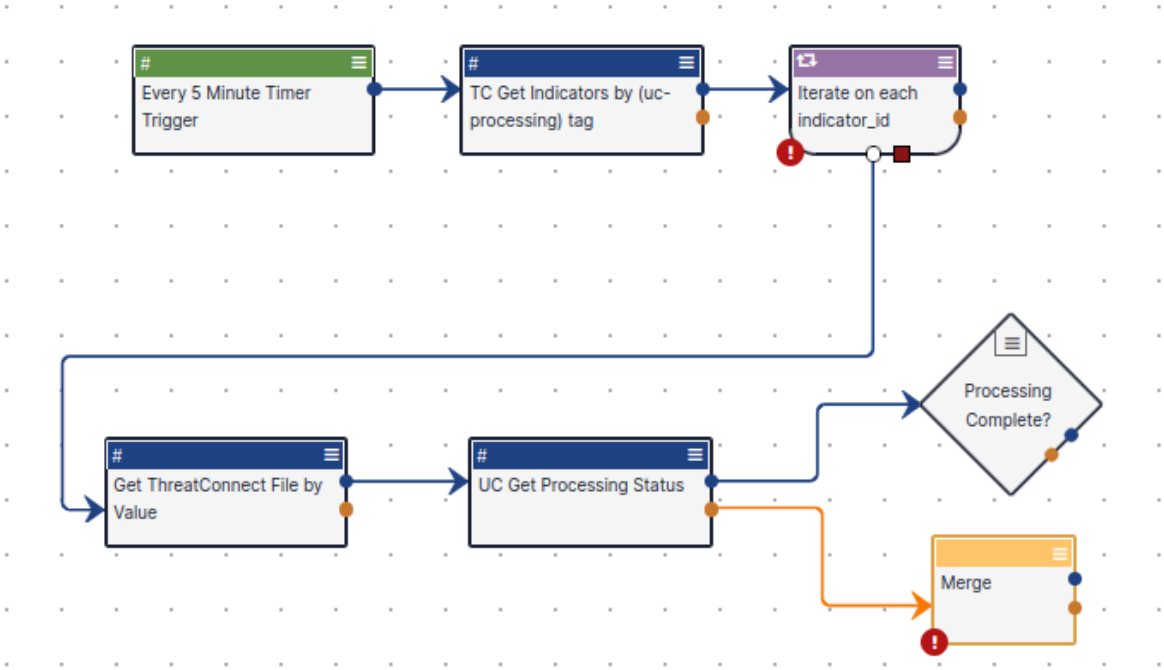
Config

Hash ID *

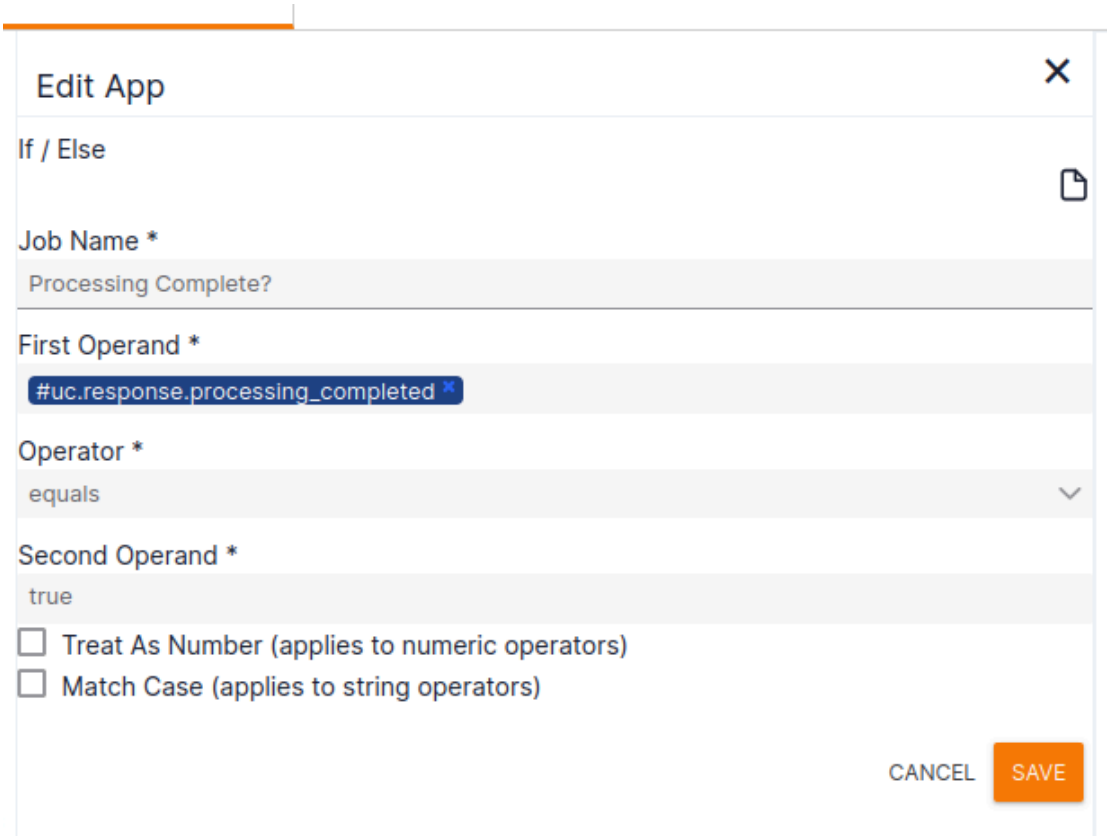
#tc.file.value

CANCEL SAVE

5. Now add an **If / Else** operator and a **Merge** operator to the playbook. Then, connect the blue, success, dot of the **Unknown Cyber** app to the **If / Else** operator and the orange, failure, dot of the **Unknown Cyber** app to the **Merge** operator. Then configure the If / Else operator as shown below.

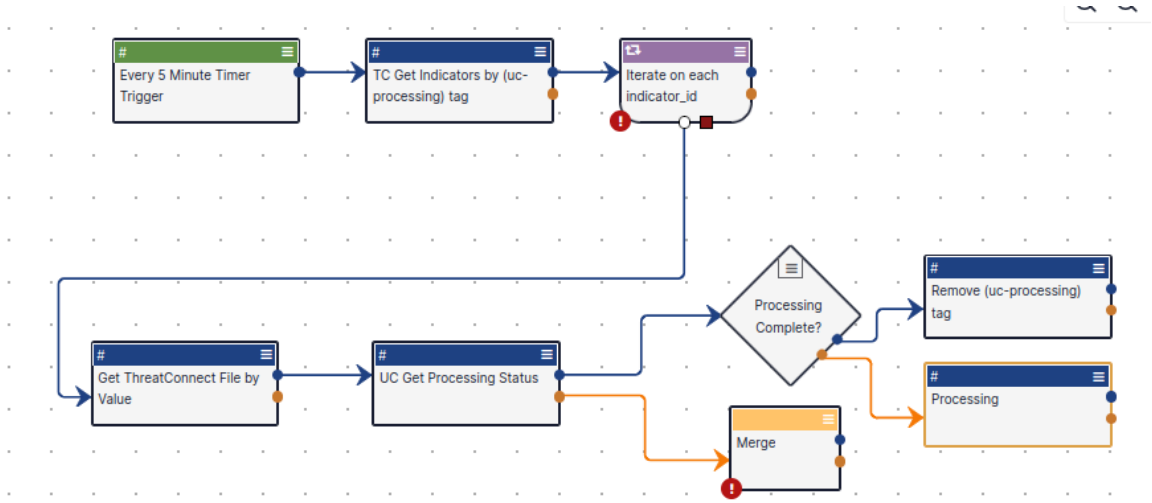


If / Else	
First Operand	#uc.response.processing_completed
Operator	equals
Second Operand	true

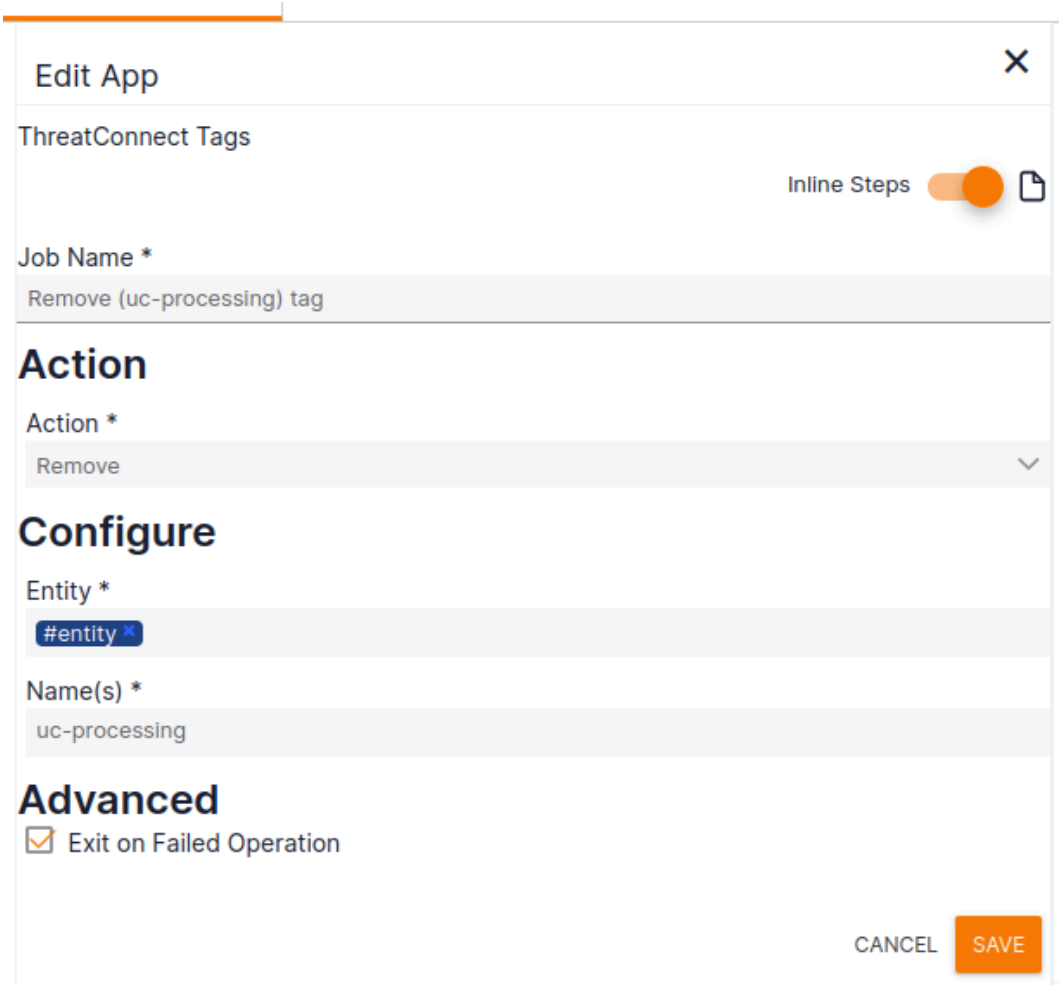


6. After the **If / Else**, add a **ThreatConnect Tags** app and **Logger** app and connect them to the success and failure dots respectively. Then configure them as shown below.

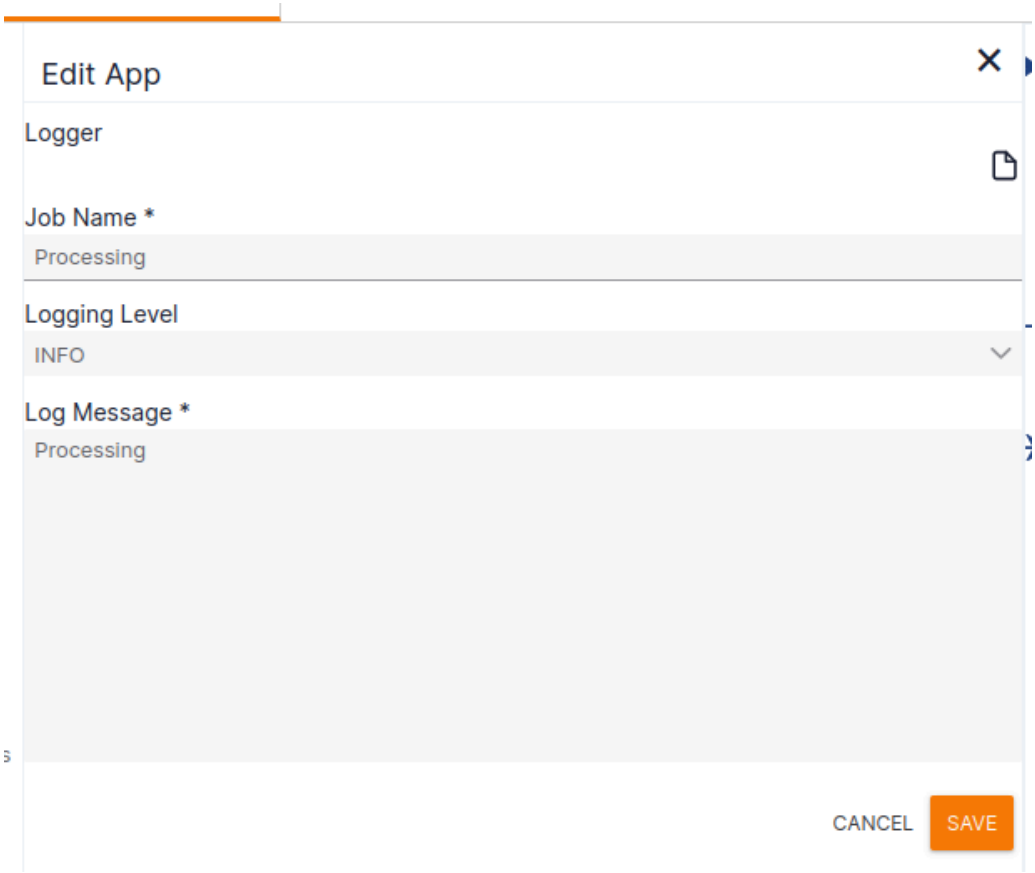
   This ThreatConnect Tags app will handle removing the *uc-processing* tag once a file has finished processing. The Logger is used to show a file is being checked, but has not finished processing.



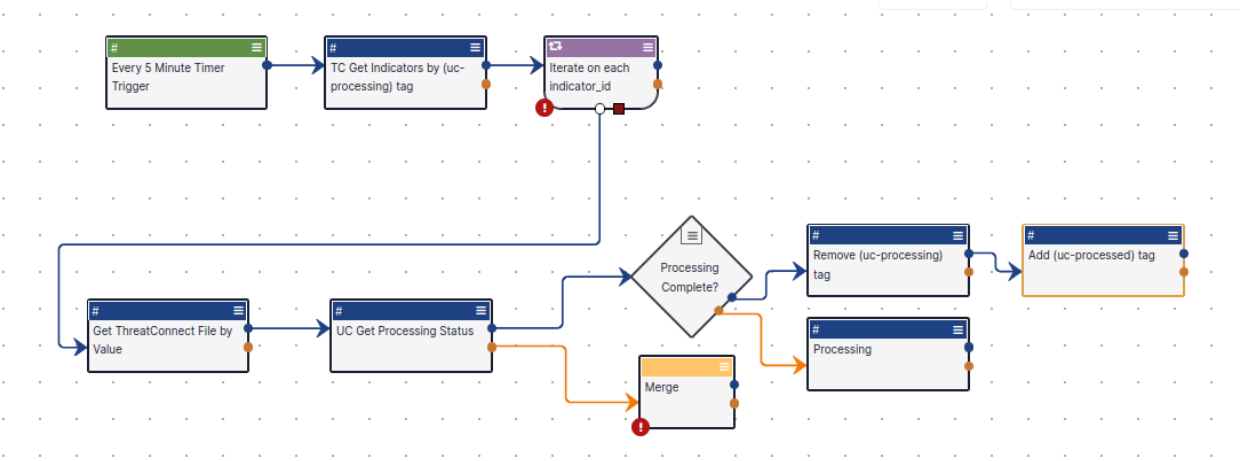
ThreatConnect Tags	
Action	Remove
Entity	#entity
Name(s)	uc-processing
Exit on Failed Operation	True



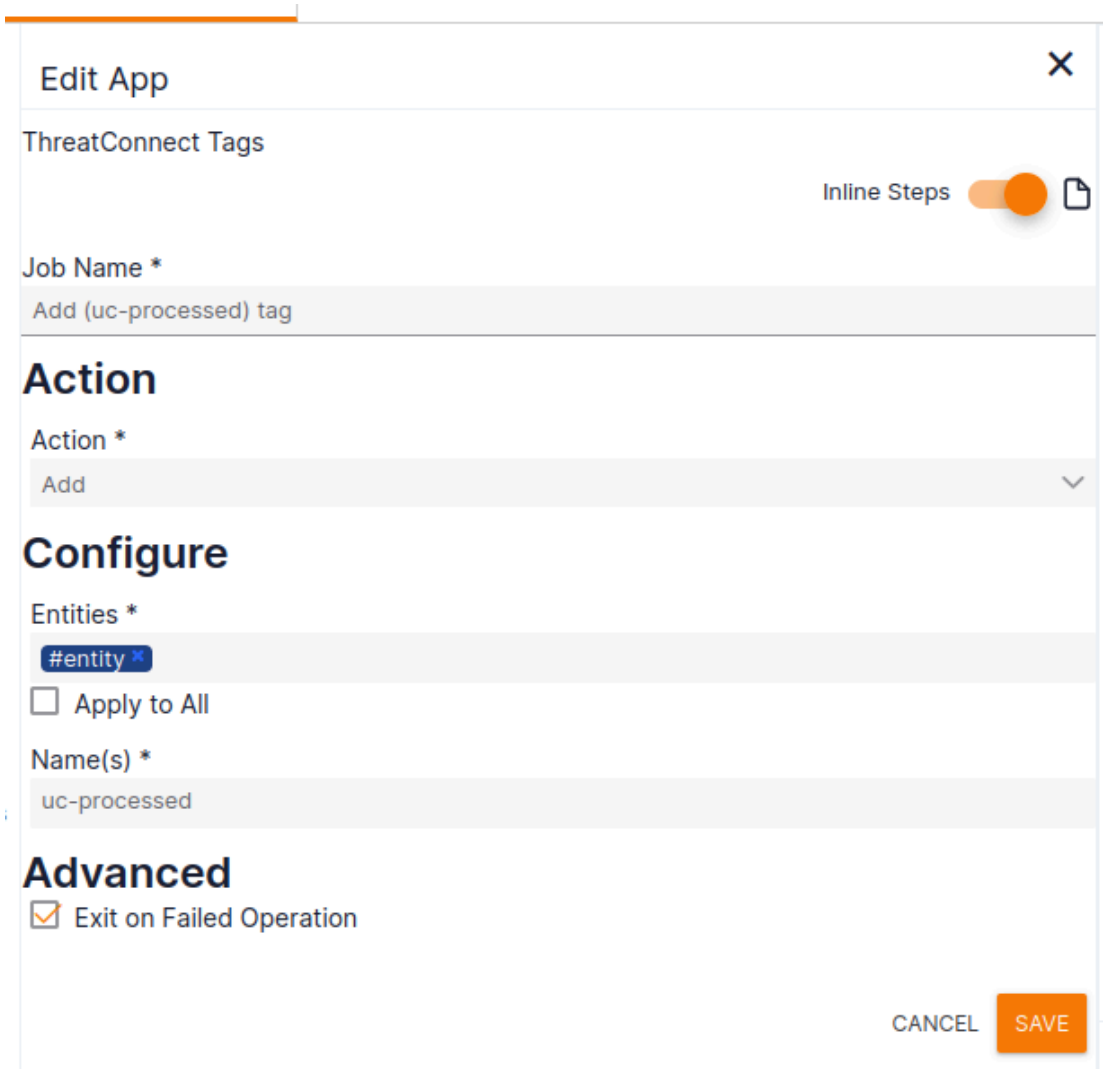
Logger	
Logging Level	Info
Log Message	Processing



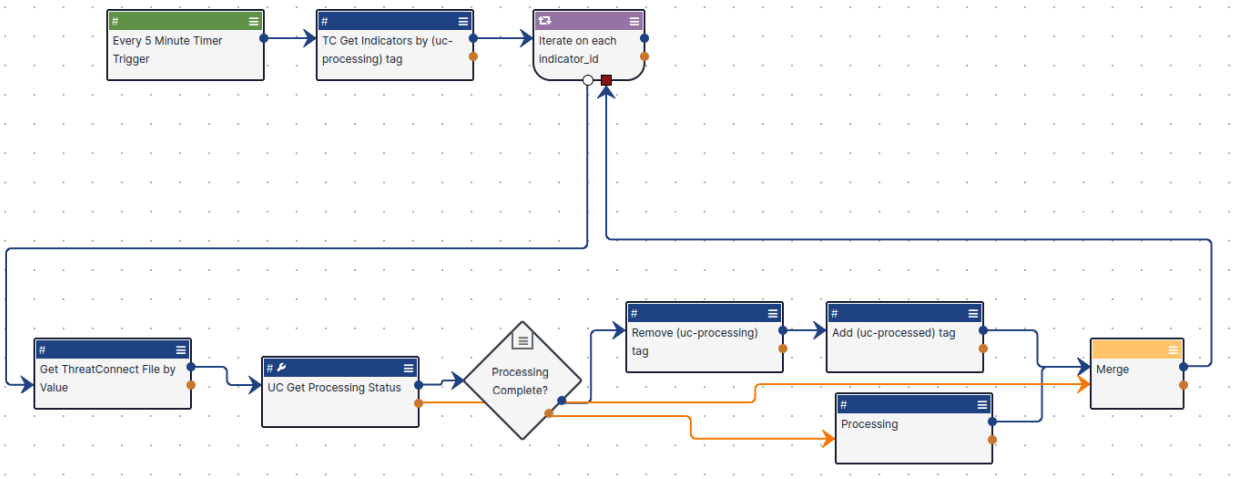
7. Now add another **ThreatConnect Tags** app and connect it behind the previous one and configure it. This one adds a *uc-processed* tag to all the File Indicators that have been processed.



ThreatConnect Tags	
Action	Add
Entity	#entity
Name(s)	uc-processed
Exit on Failed Operation	True



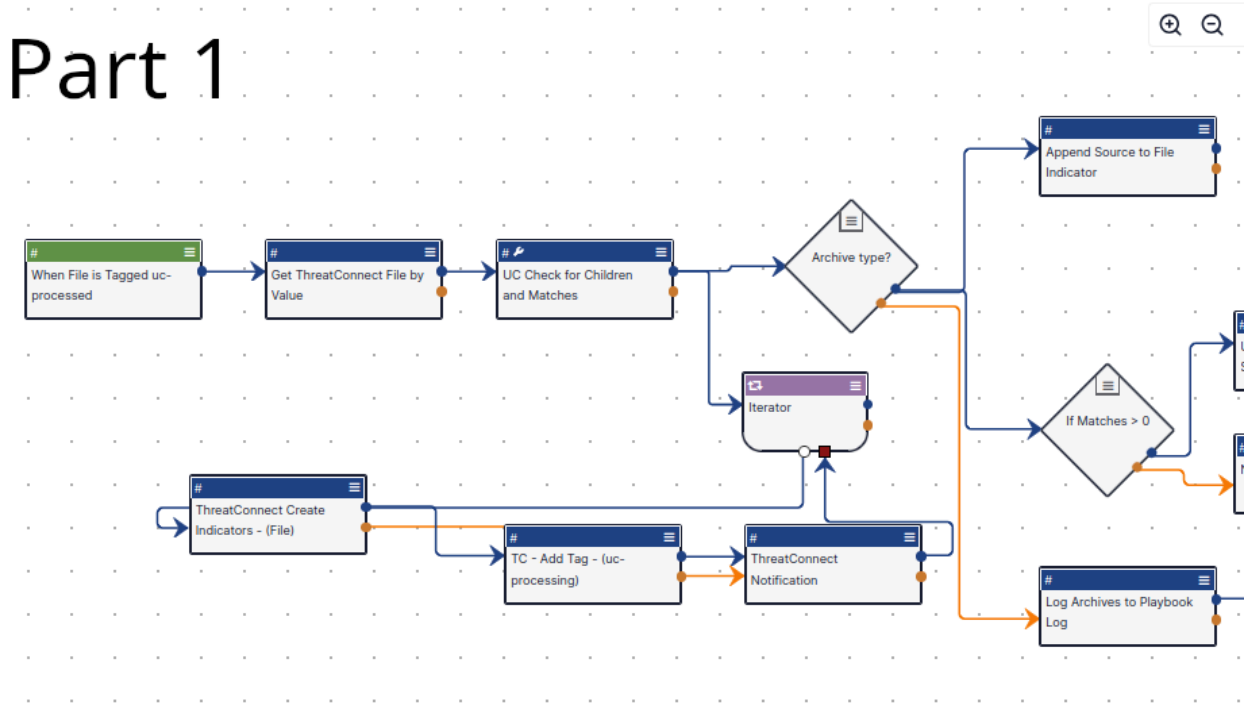
8. Lastly, connect the success of the **ThreatConnect Tags (Add)** and **Logging** to the **Merge**. Then connect the merge to the red square (closing the loop) of the **Iterator**.



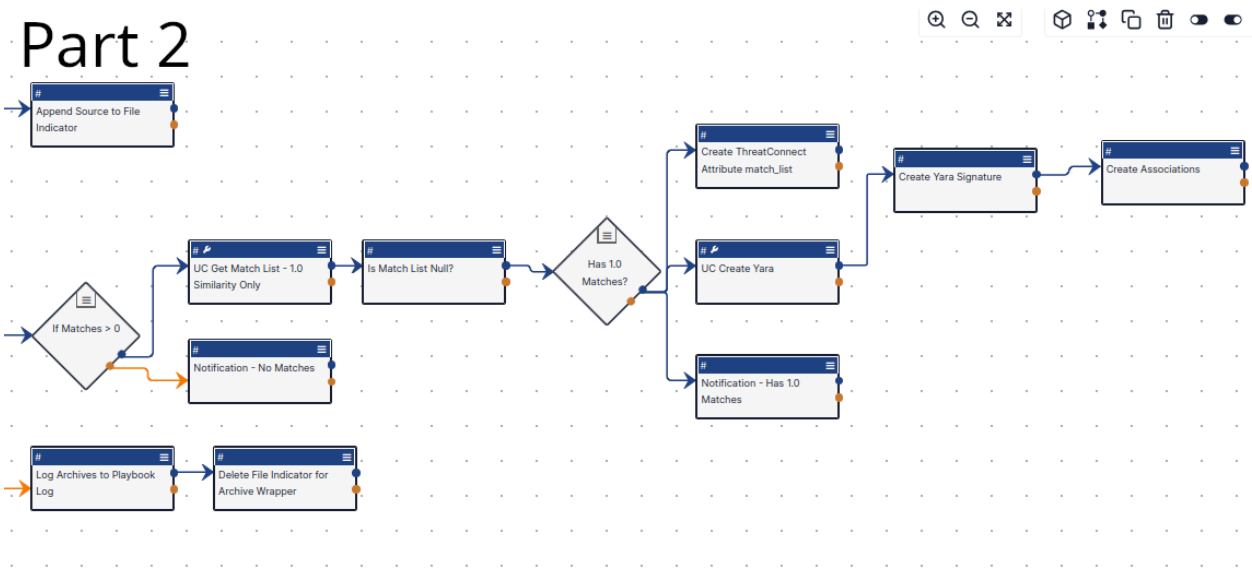
5.3 Get Matches

This playbook is the third out of three. This playbook runs once a File Indicator is tagged with *uc-processed* and adds detailed insights to the Indicator. Insights include how many other files have a 1.0 match, which can be configured down to a 0.7 match, and if there are matches, a yara rule will automatically be generated and attributed to the File Indicator. Also, if the file has children, which most archives will, then it will append a *uc-processing* tag to each of the children to track when they are done processing.

Note: The playbook is long, so the overview pictures are split into **Part 1** (left) and **Part** 2 (right).



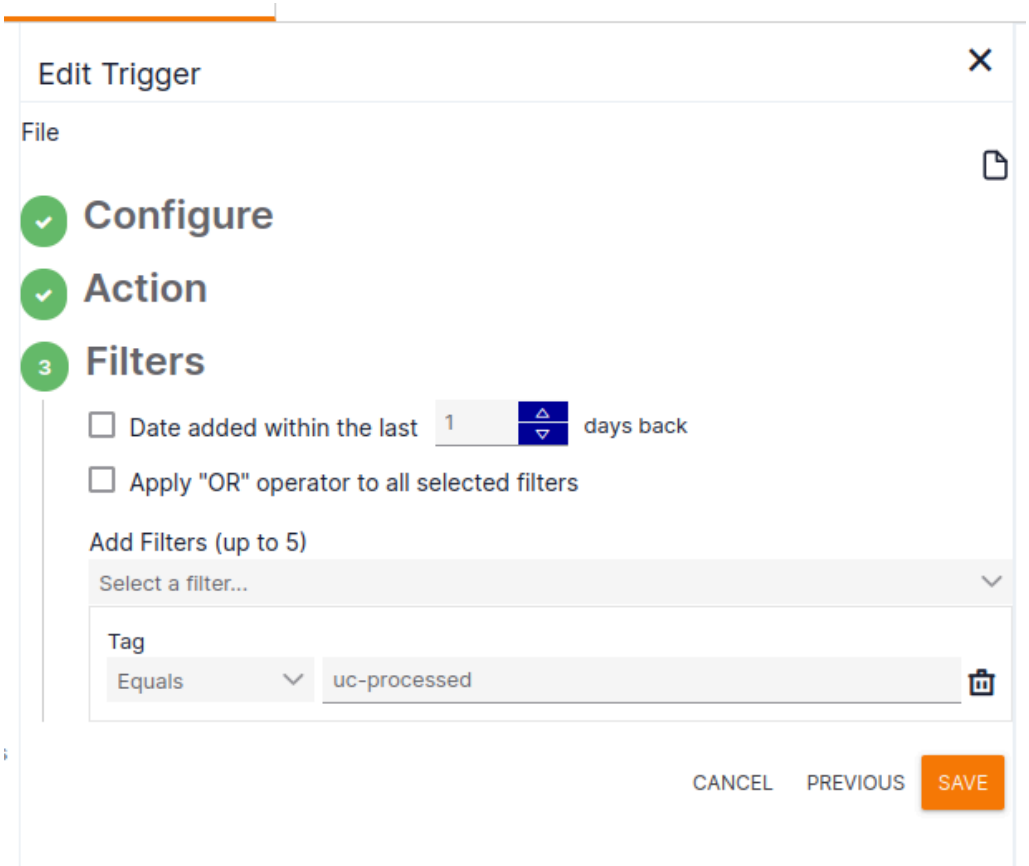
Part 2



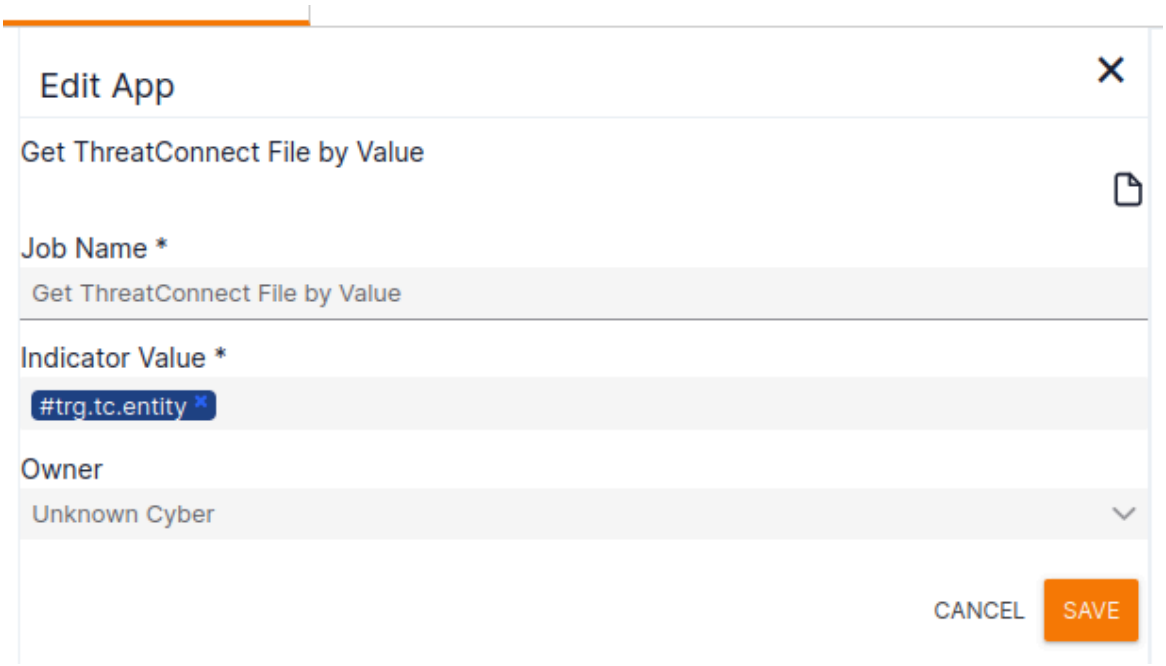
Steps

1. The playbook starts with a **File** Indicator Trigger which is connected to a **Get ThreatConnect File by Value** app. See the configuration for both apps below.

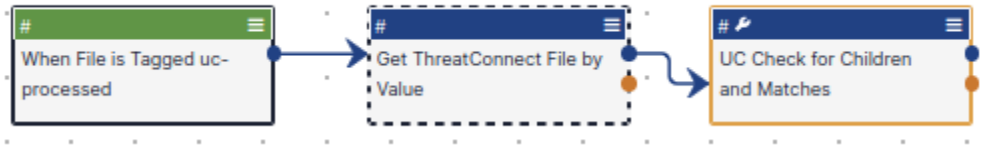
File (Trigger)	
Owners	Which Intel Source you want to trigger the playbook.
Action Type	Tag Applied
Filters - Select Tag	
Equals	uc-processed



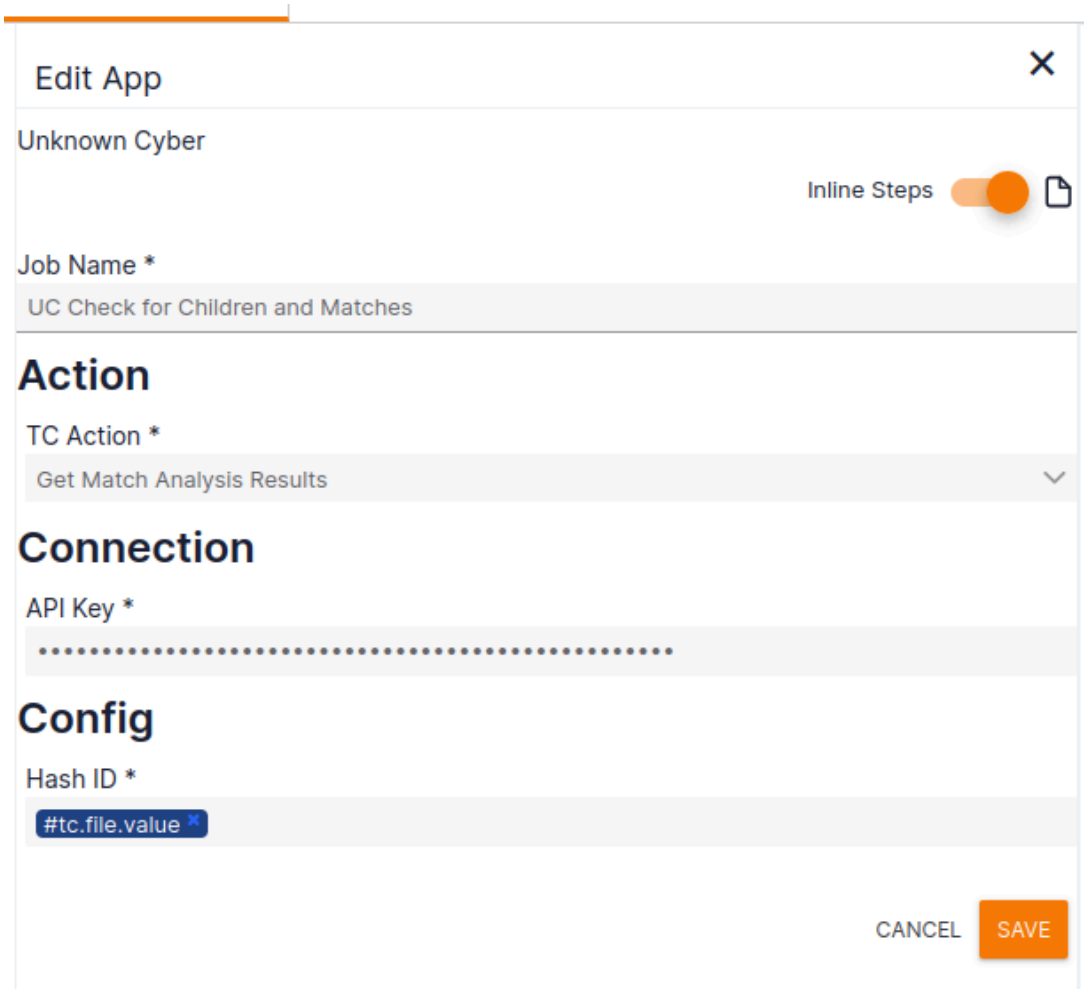
Get ThreatConnect File by Value	
Indicator Value	#trg.tc.entity
Owner	Which Intel Source you want to trigger the playbook.



2. Now add the **Unknown Cyber** app and connect it behind **Get ThreatConnect File by Value**. Then configure it as shown below.

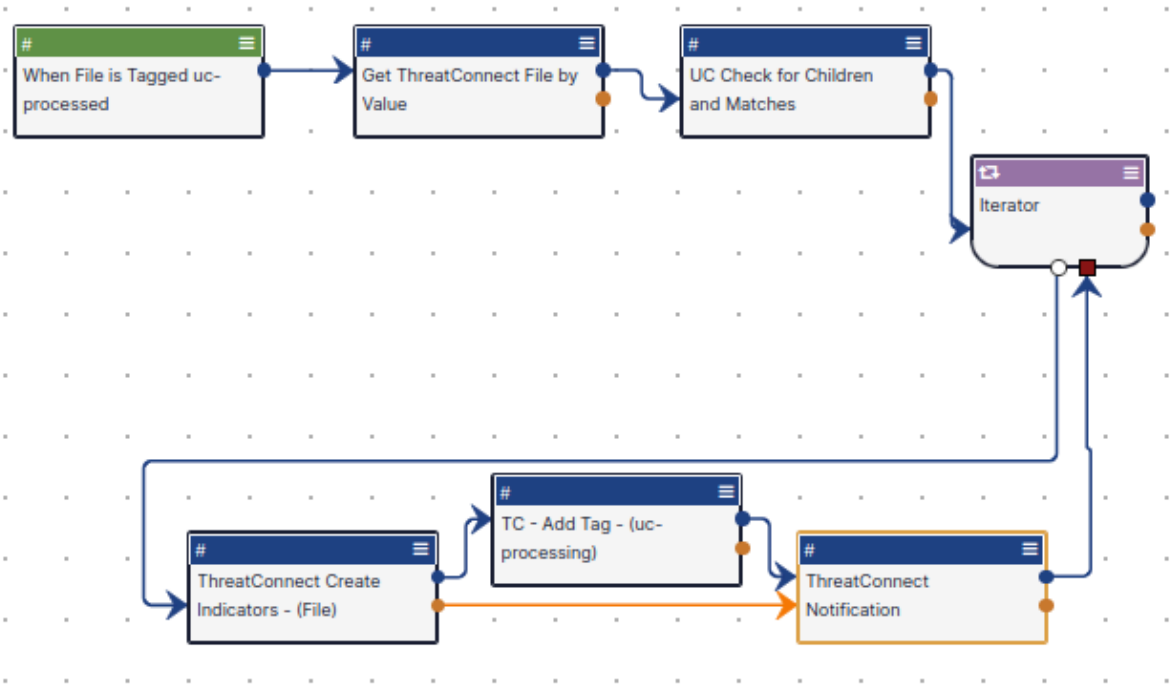


Unknown Cyber (Get Match Analysis Results)	
TC Action	Get Match Analysis Results
API Key	Your Unknown Cyber API Key
Hash ID	#tc.file.value

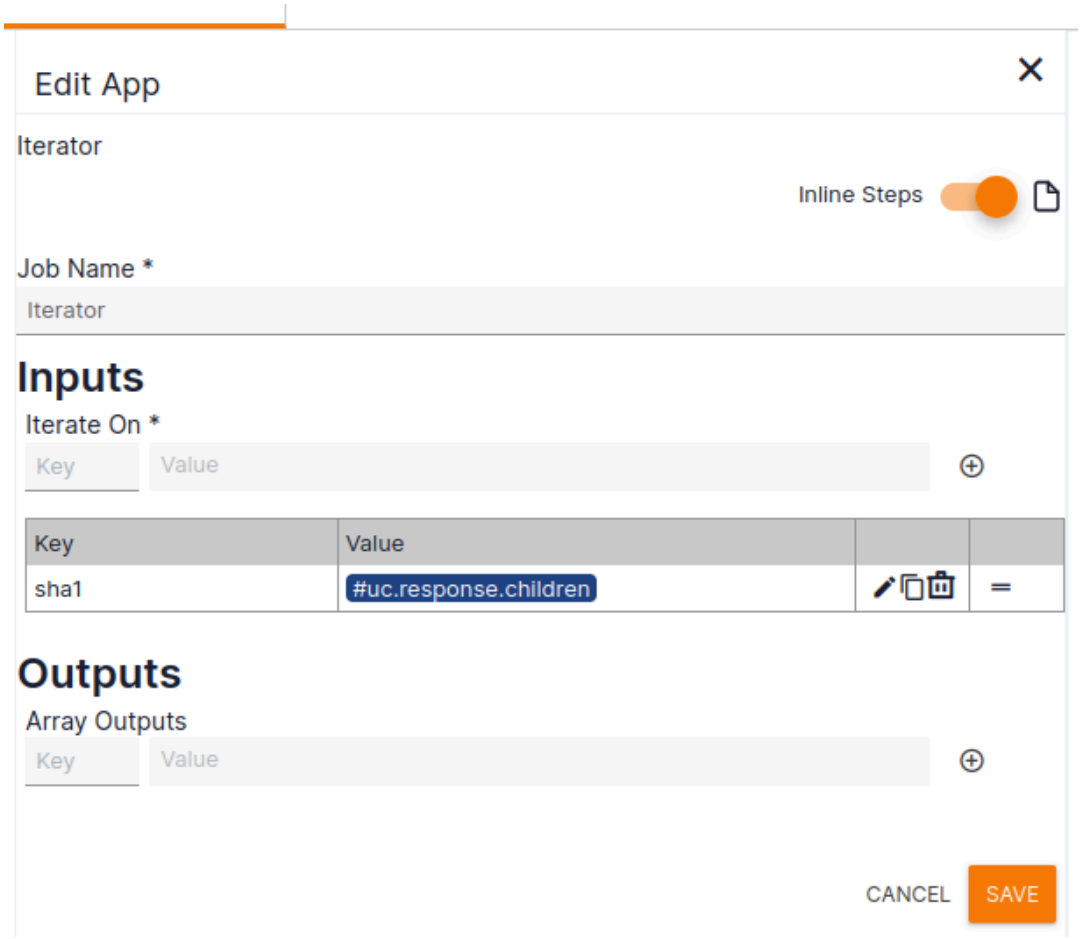


3. Now, add an **Iterator** operator and connect it to the **Unknown Cyber** app. Then, from the start of the iterator, connect these three apps, in order, using their success path: **ThreatConnect Create Indicators**, **ThreatConnect Tags**, and **ThreatConnect Notification**. After, connect the success of the ThreatConnect Notification to the red box (completing the loop) of the Iterator. Lastly, connect the failure of **ThreatConnect Create Indicators** to the **ThreatConnect Notification** app. Then configure all four of these as shown below.

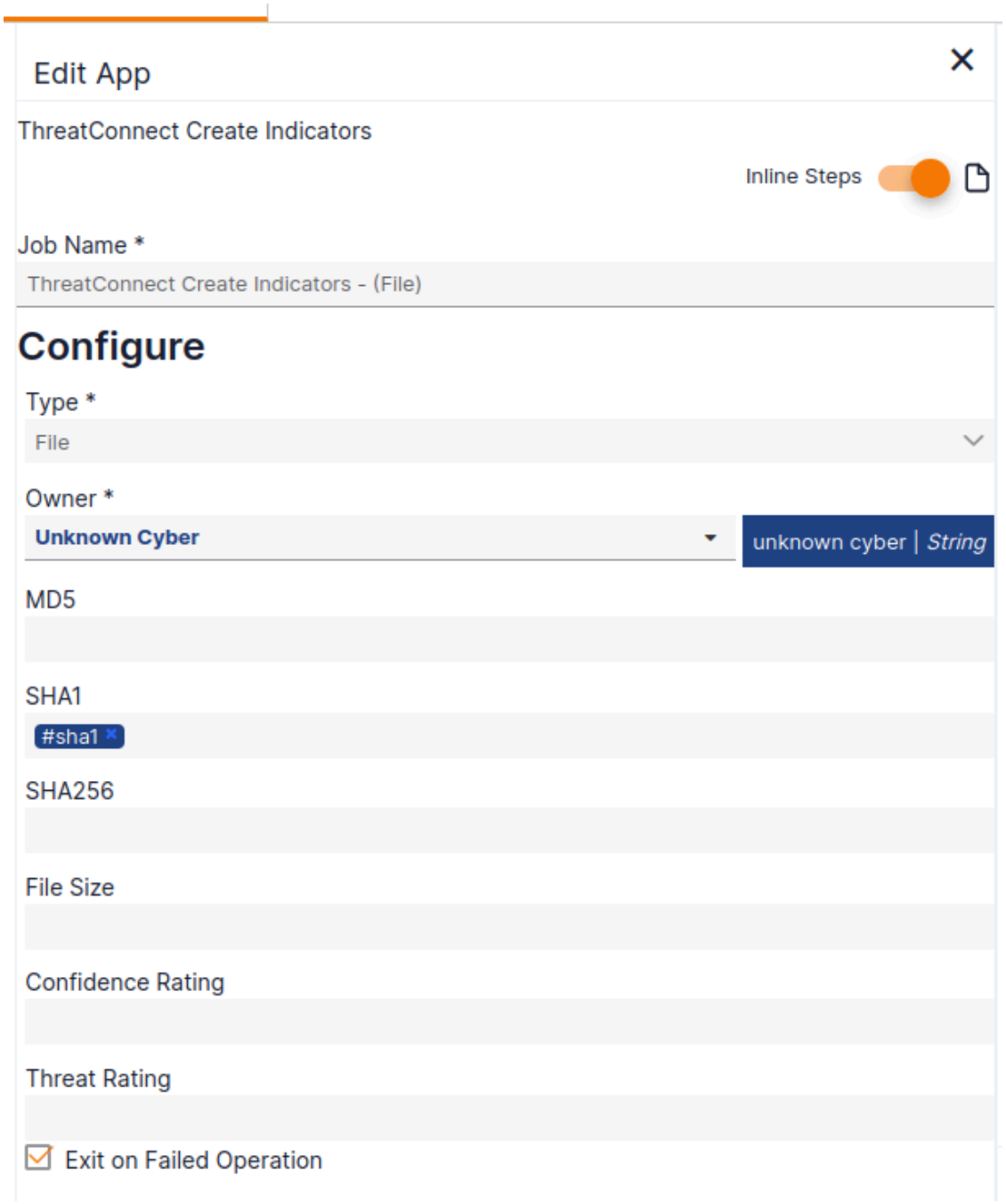
   This loop is used to recursively create File Indicators for any potential children the uploaded item had. The children are returned from Unknown Cyber's app.



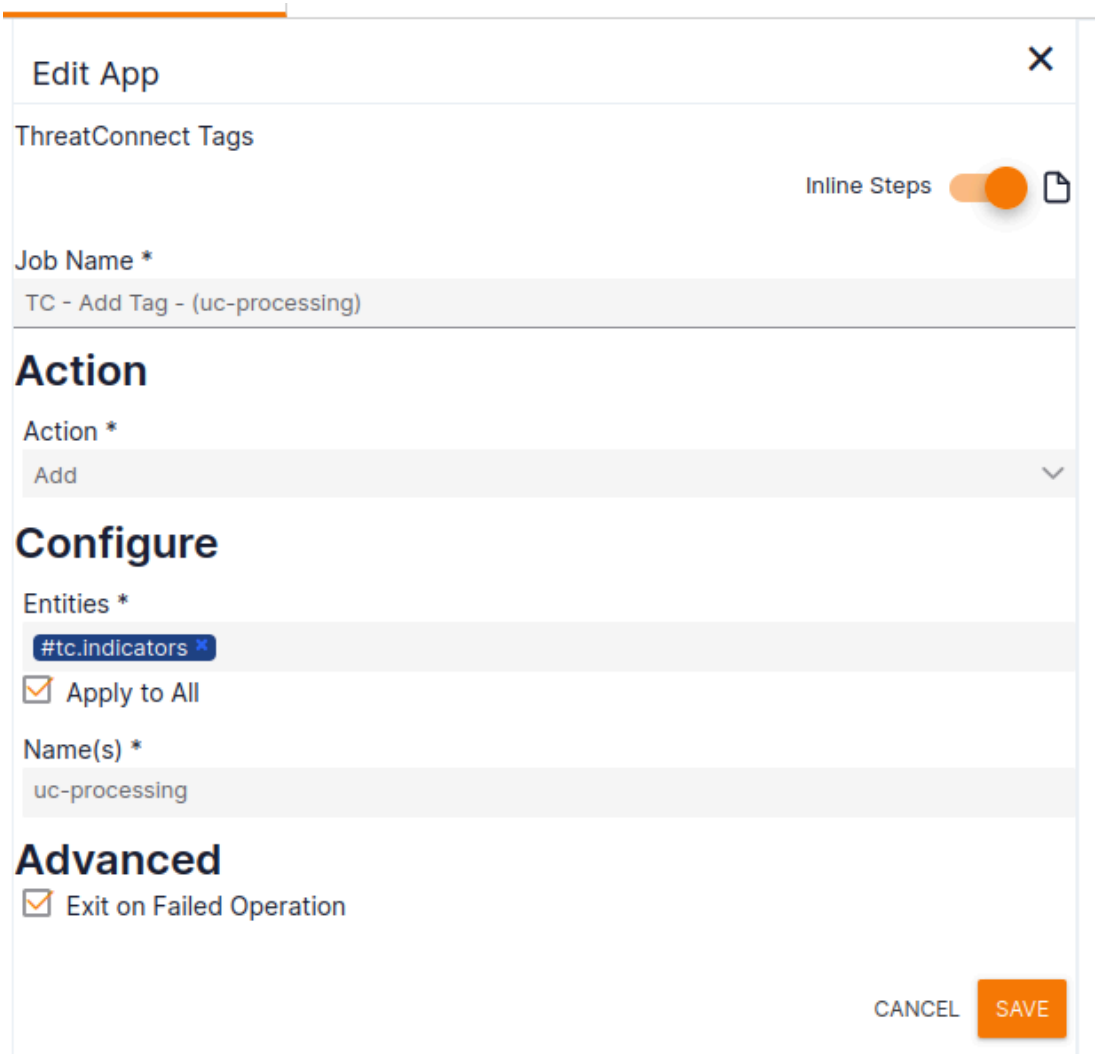
Iterator	
Inputs	
Key	sha1
Value	#uc.response.children



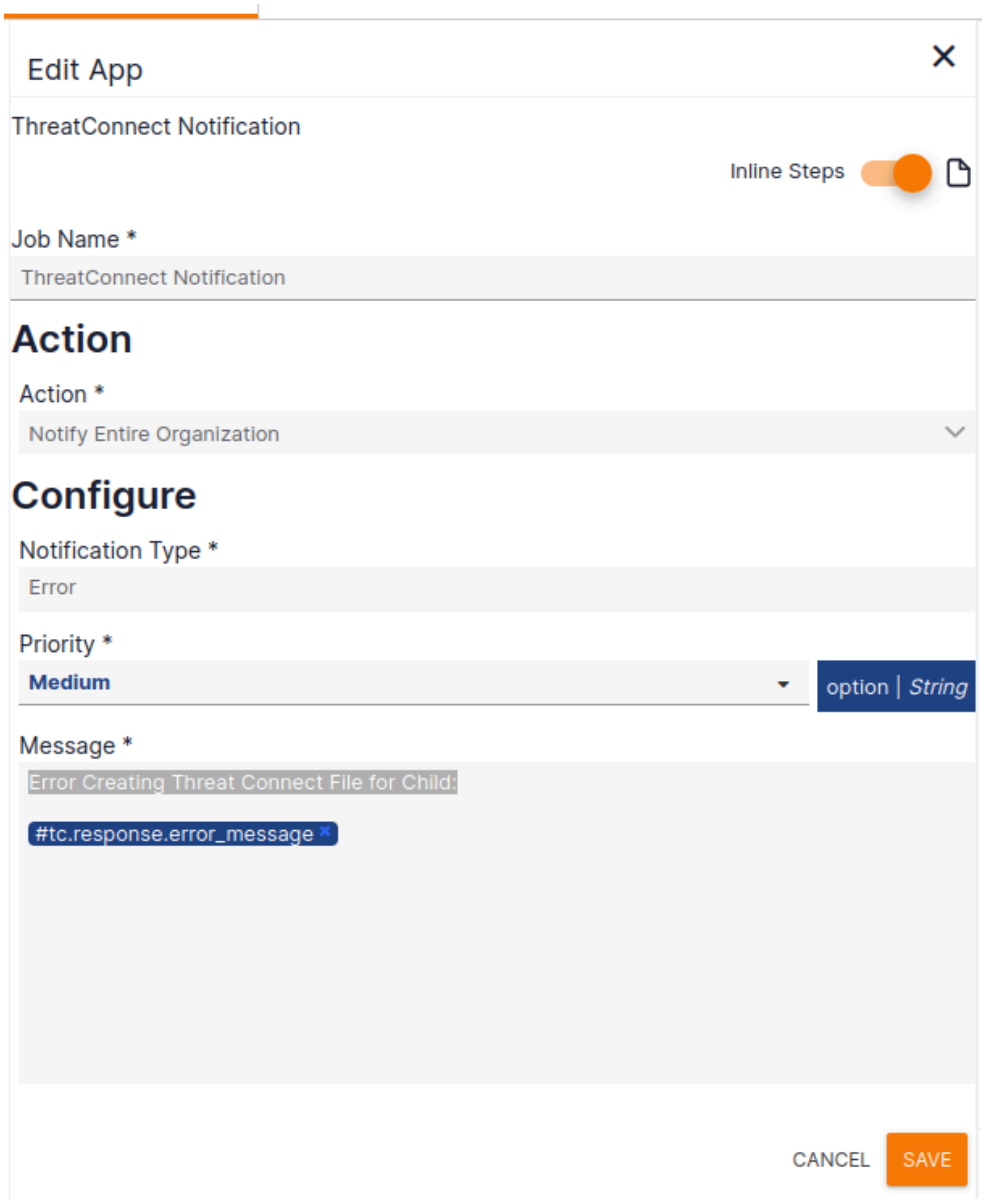
ThreatConnect Create Indicators	
Type	File
Owner	Which Intel Source you want to trigger the playbook.
SHA1	#sha1
Exit of Failed Operation	True



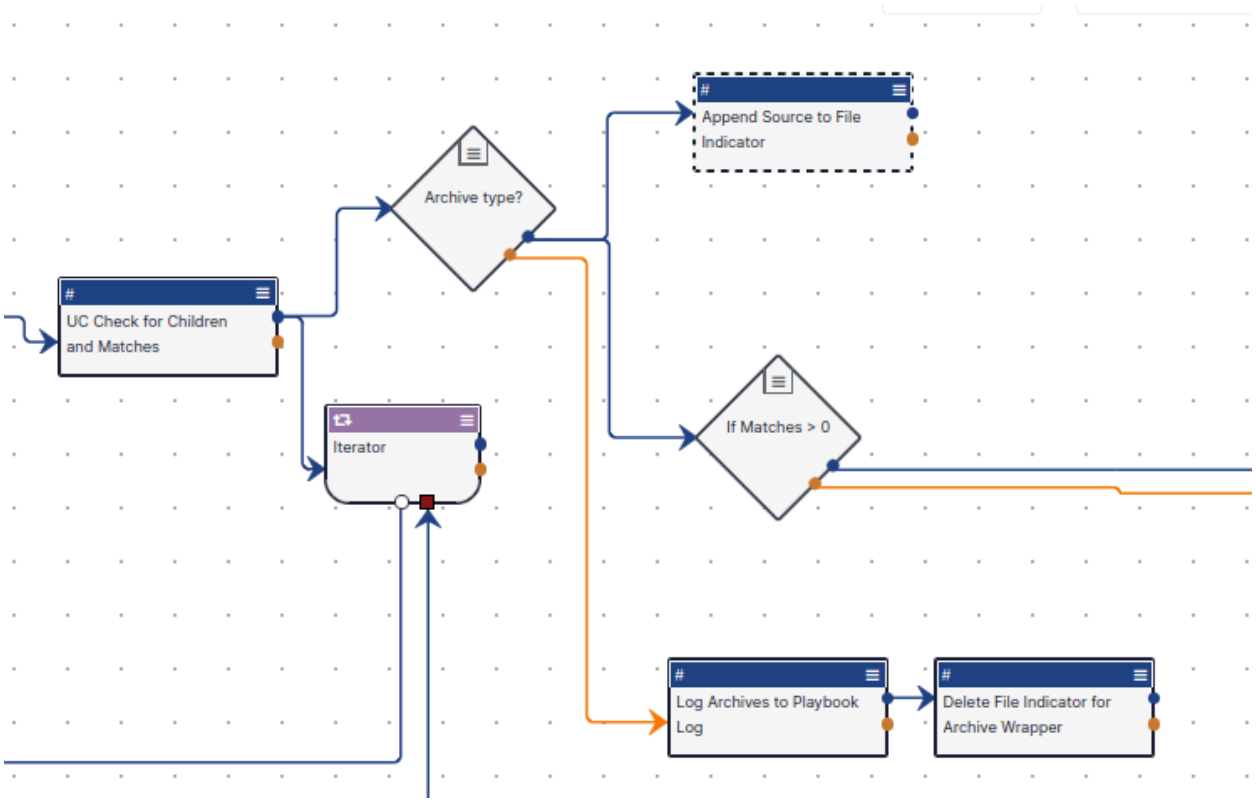
ThreatConnect Tags	
Action	Add
Entities	#tc.indicators
Apply to All	True
Name(s)	uc-processing
Exit of Failed Operation	True



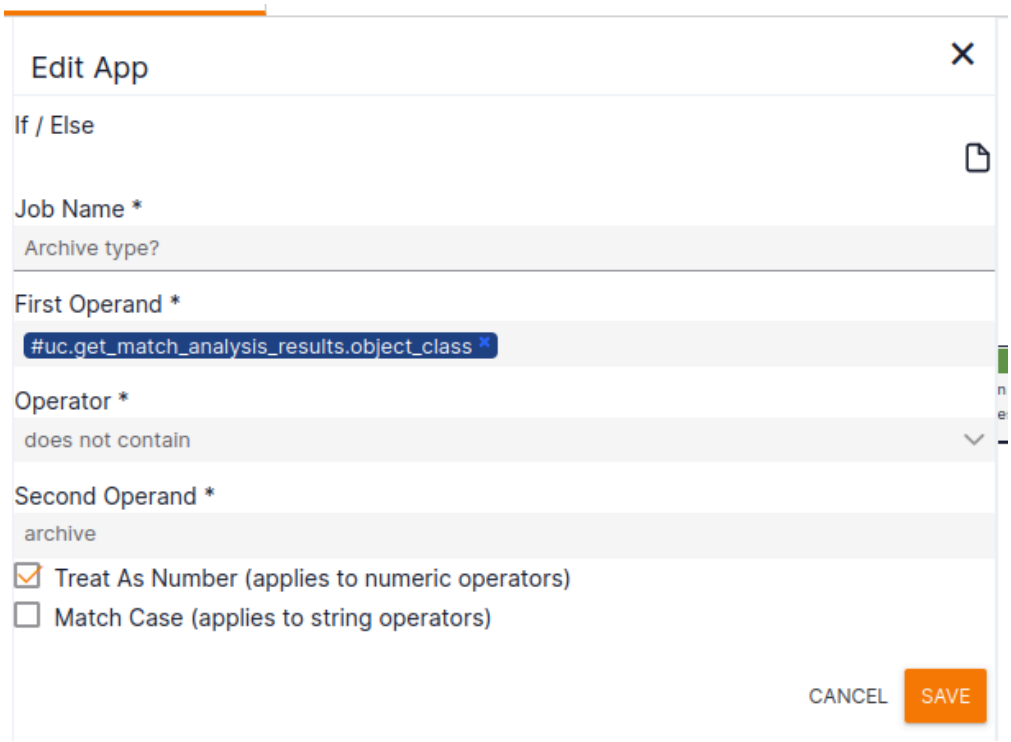
ThreatConnect Notification	
Action	Notify Entire Organization
Notification Type	Error
Priority	Medium
Message	Error Creating ThreatConnect File for Child: #tc.response.error_message
Exit of Failed Operation	True



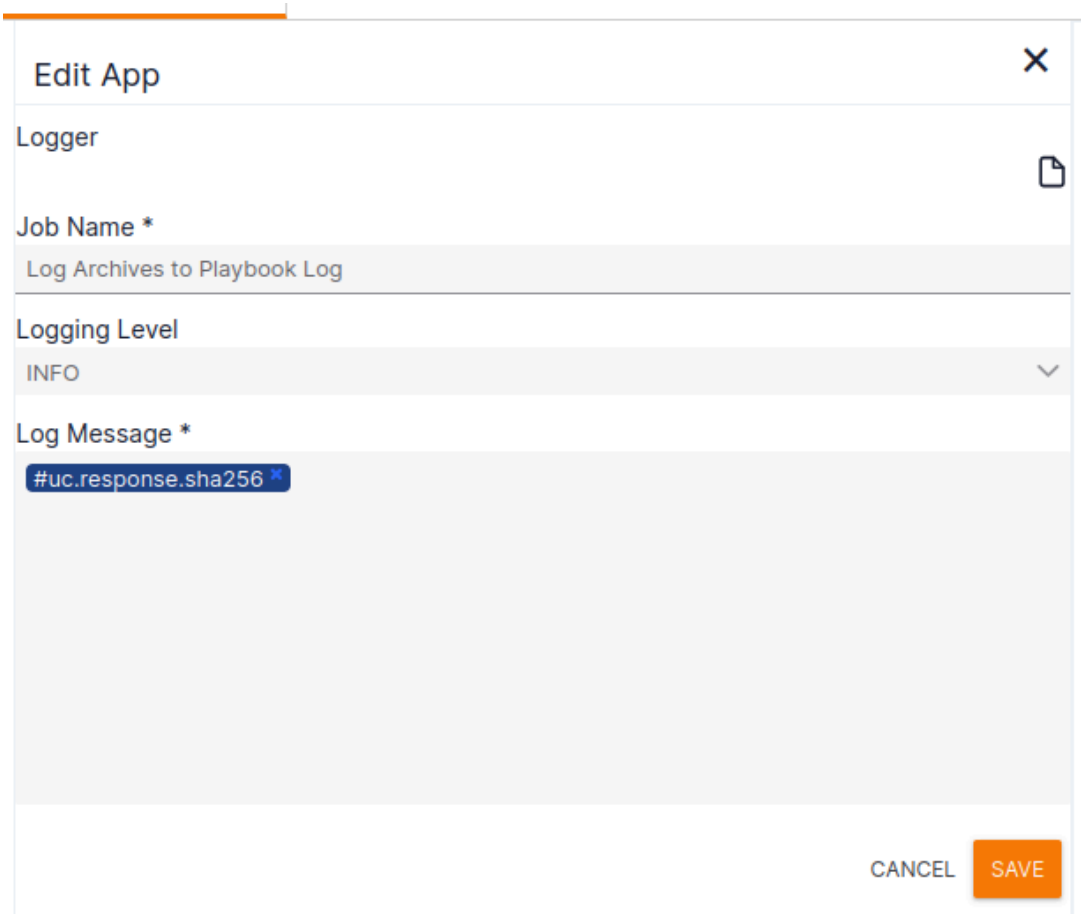
4. While the Iterator loop creates File Indicators for all the files children, if any, an **If / Else** operator will check to see if the *object_class* of the File Indicator is an **archive**. If it is, the **Else** will run, log the archive using a **Logger** app, and then delete the Indicator using a **ThreatConnect Delete Indicators** app. (This can be removed if you need the archive wrapper to stay as an indicator.) If the File Indicator is Not an archive, the source, a link leading to the resource on www.unknowncyber.com, is appended to the Indicator using **Create ThreatConnect Attribute** app and another **If / Else** check is made to determine if the File Indicator matches any other file on Unknown Cyber.



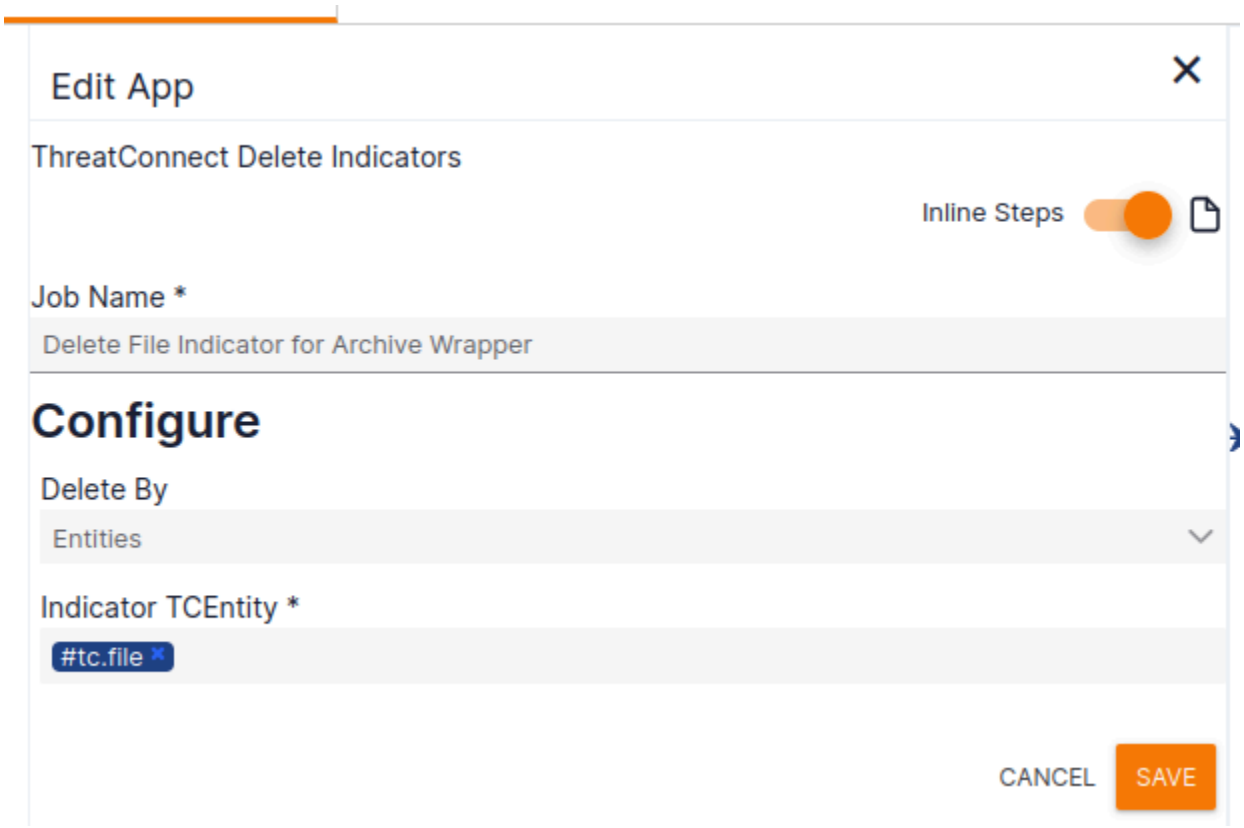
If / Else (Archive Check)	
First Operand	#uc.get_match_analysis_results.object_class
Operator	does not contain
Second Operand	archive
Treat as Number	True



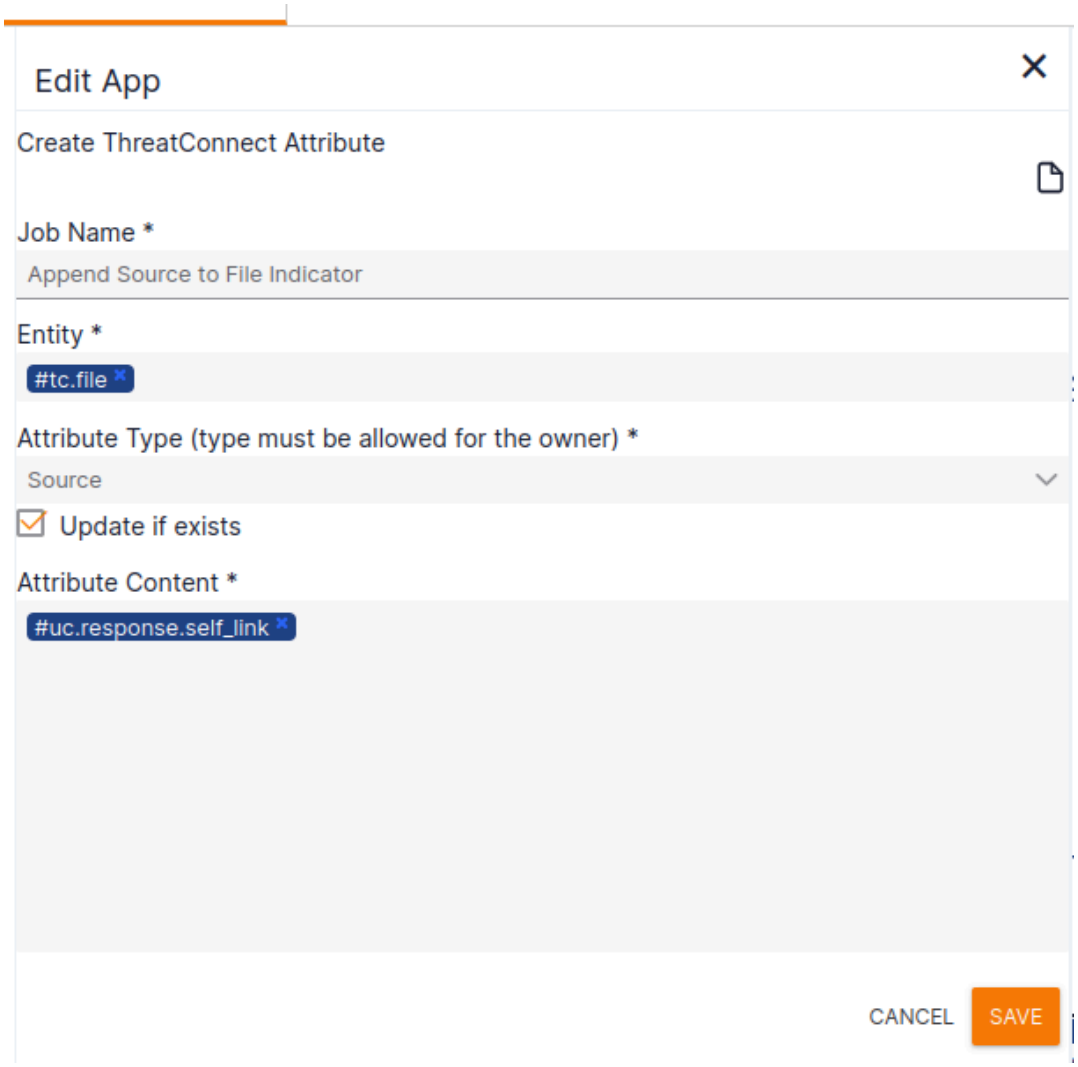
Logger	
Logging Level	INFO
Log Message	#uc.response.sha256



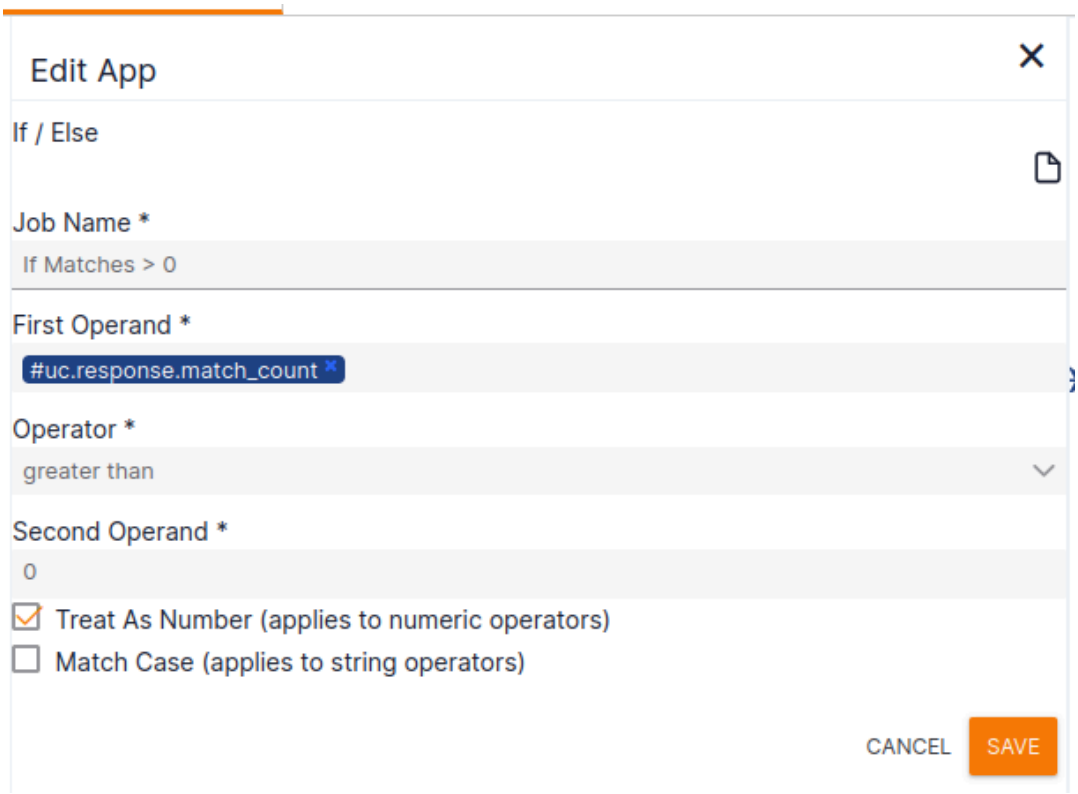
ThreatConnect Delete Indicators	
Delete By	Entities
Indicator TCEntity	#tc.file



Create ThreatConnect Attribute	
Entity	#tc.file
Attribute Type	Source
Update if exists	True
Attribute Content	#uc.response.self_link

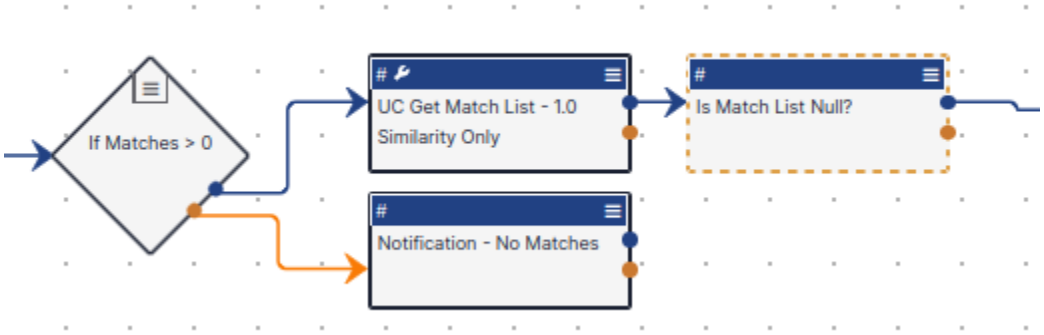


If / Else (Matches Count)	
First Operand	#uc.response.match_count
Operator	greater than
Second Operand	0
Treat as Number	True

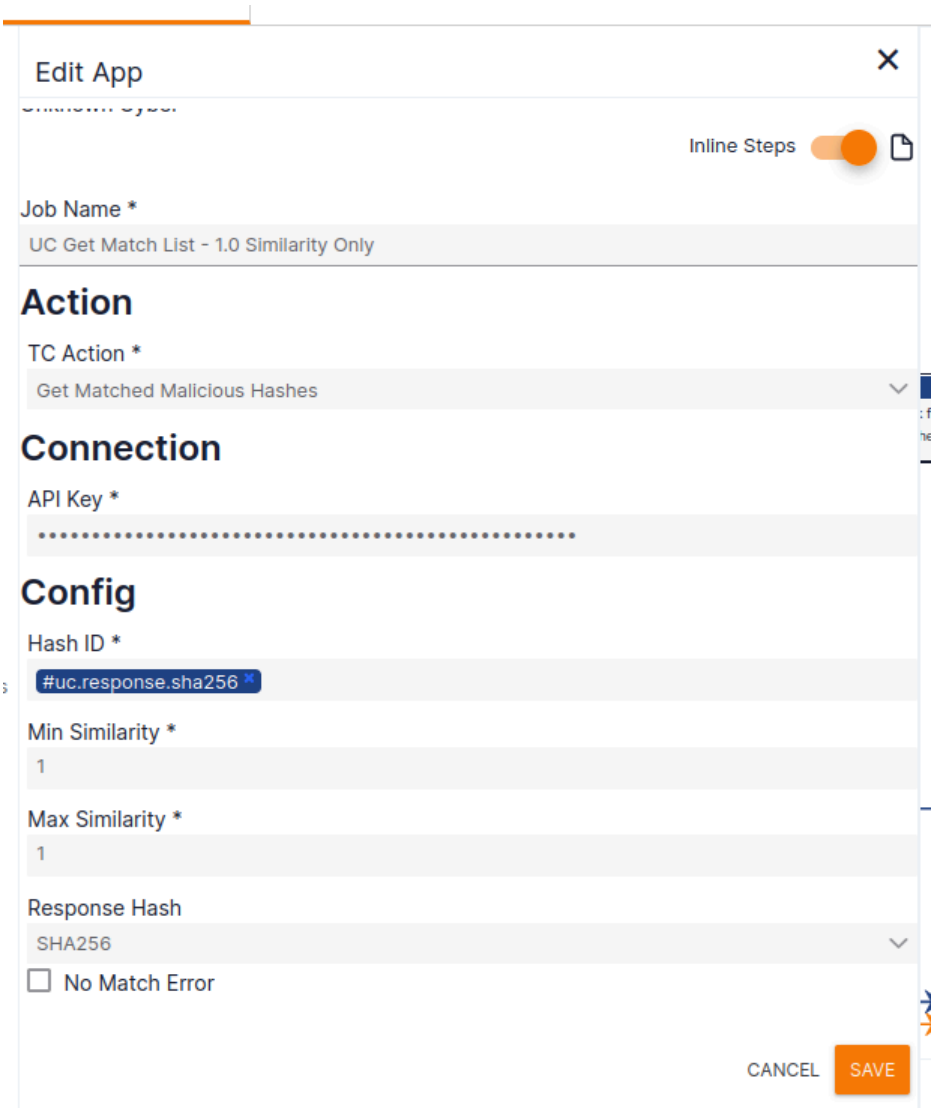


5. Continuing from the **If / Else** operator that checks to see if a file has any matches, there are three apps. The first one is the **Unknown Cyber** app and it connects behind the successful **If / Else** check. Then, a **String Operations** app is connected to **Unknown Cyber** and is used to check if the list of similarities returned by Unknown Cyber is null / empty or not.

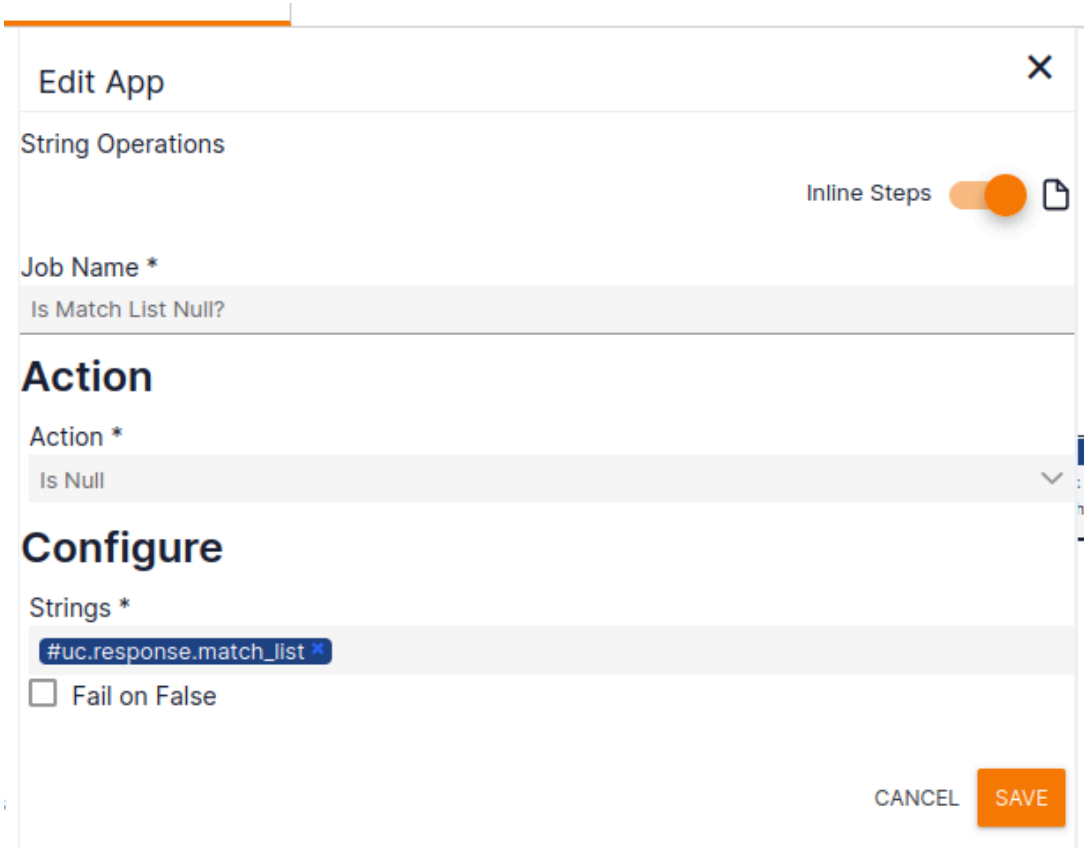
   Coming from the else portion of the **If / Else**, a **ThreatConnect Notification** app is used to notify you there are no matches. This portion can be omitted.



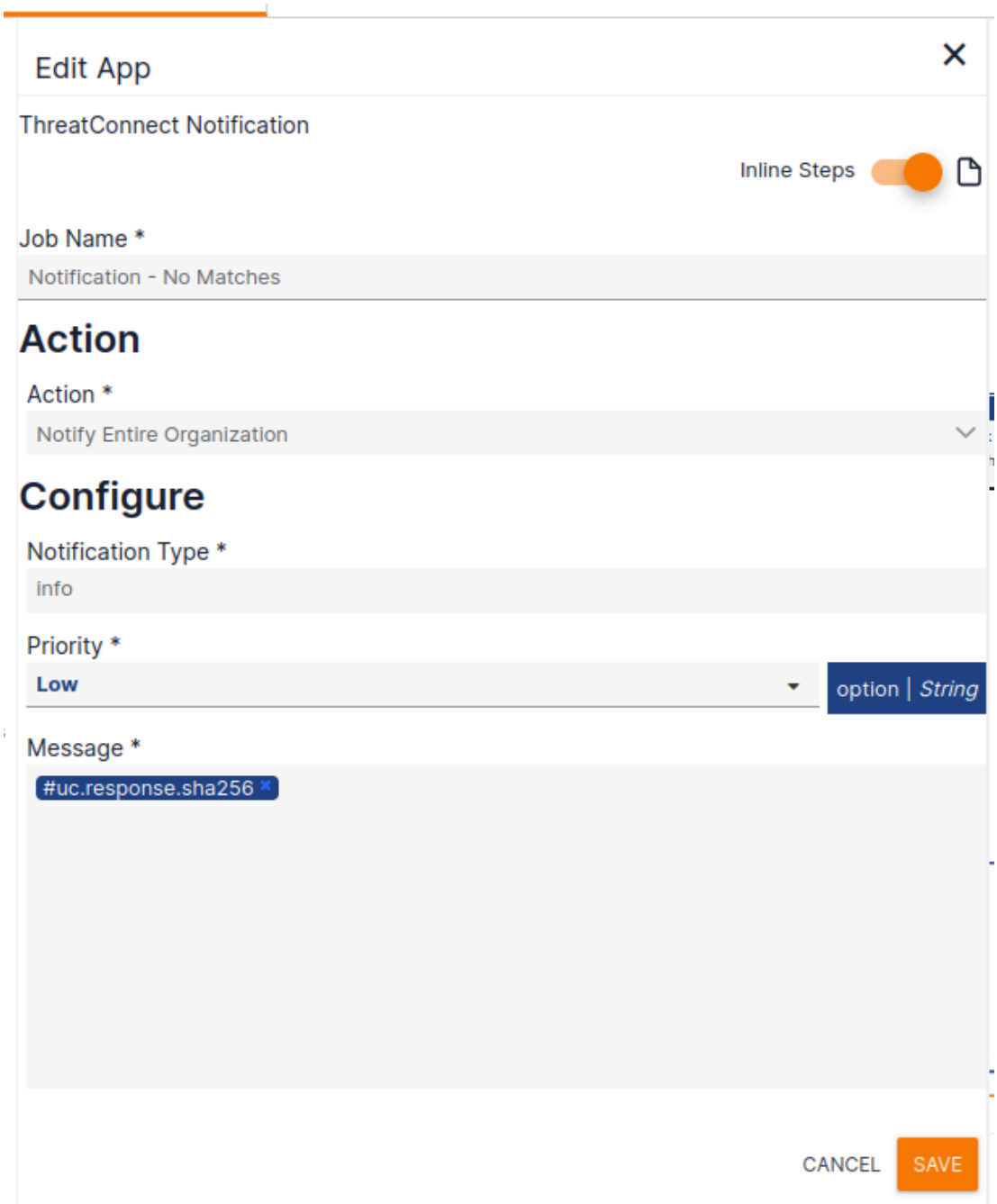
Unknown Cyber (Get Matched Malicious Hashes)	
TC Action	Get Matched Malicious Hashes
API Key	Your Unknown Cyber API Key
Hash ID	#uc.response.sha256
Min Similarity	1 **Note:** For the example, we are only matching 1.0 matches.
Max Similarity	1
Response Hash	SHA256



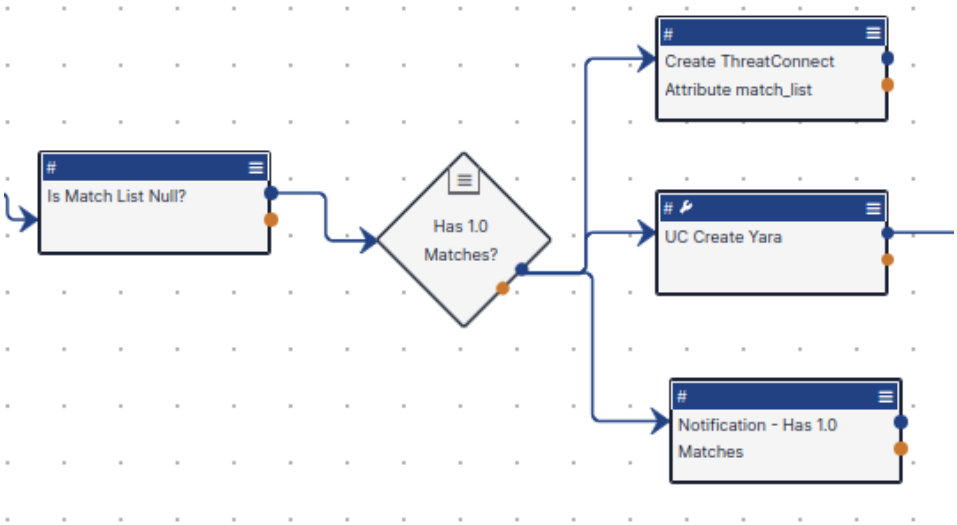
String Operations	
Action	Is Null
Strings	#uc.response.match_list



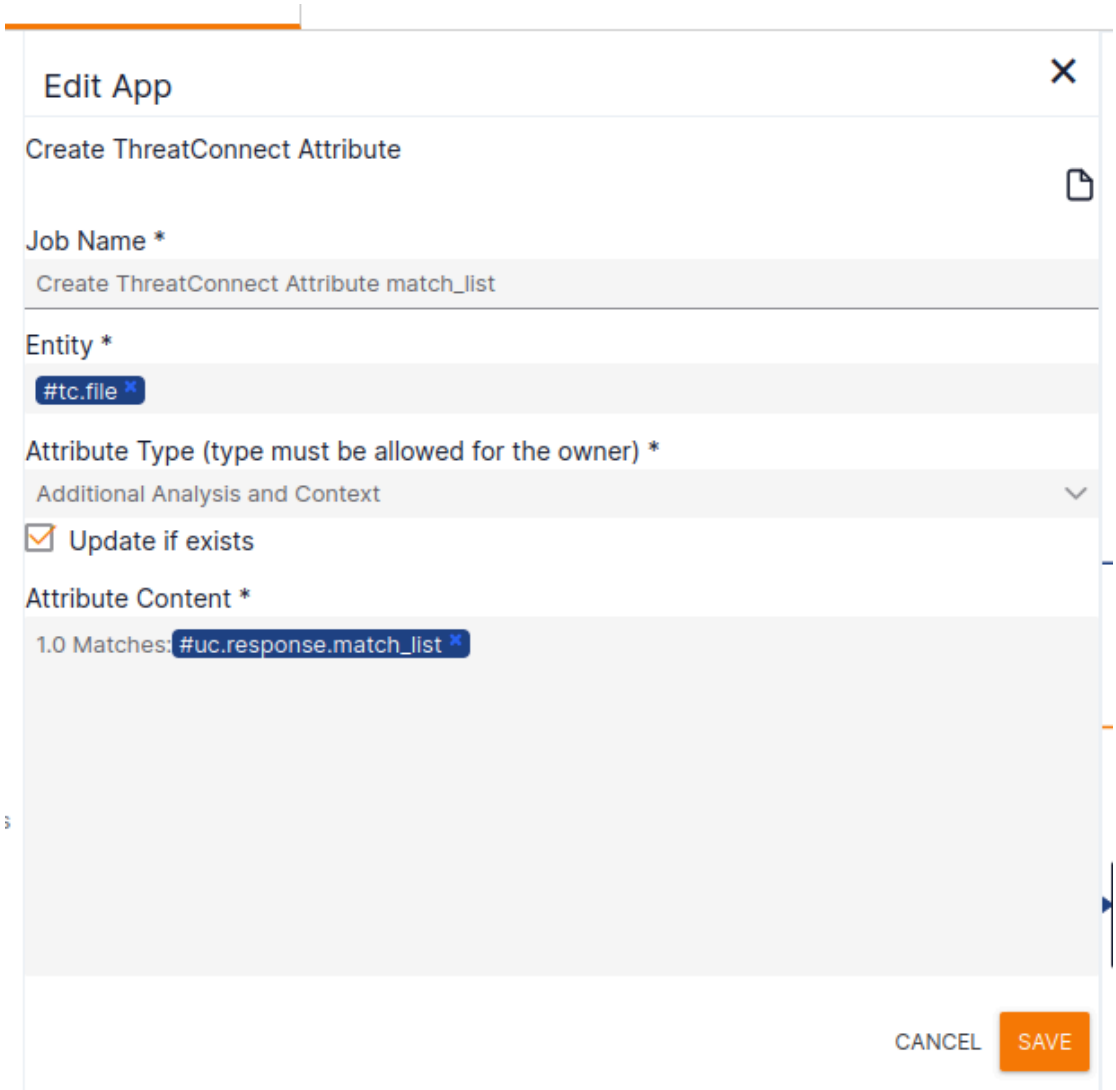
ThreatConnect Notification	
Action	Notify Entire Organization
Notification Type	Info
Priority	Low
Message	#uc.response.sha256



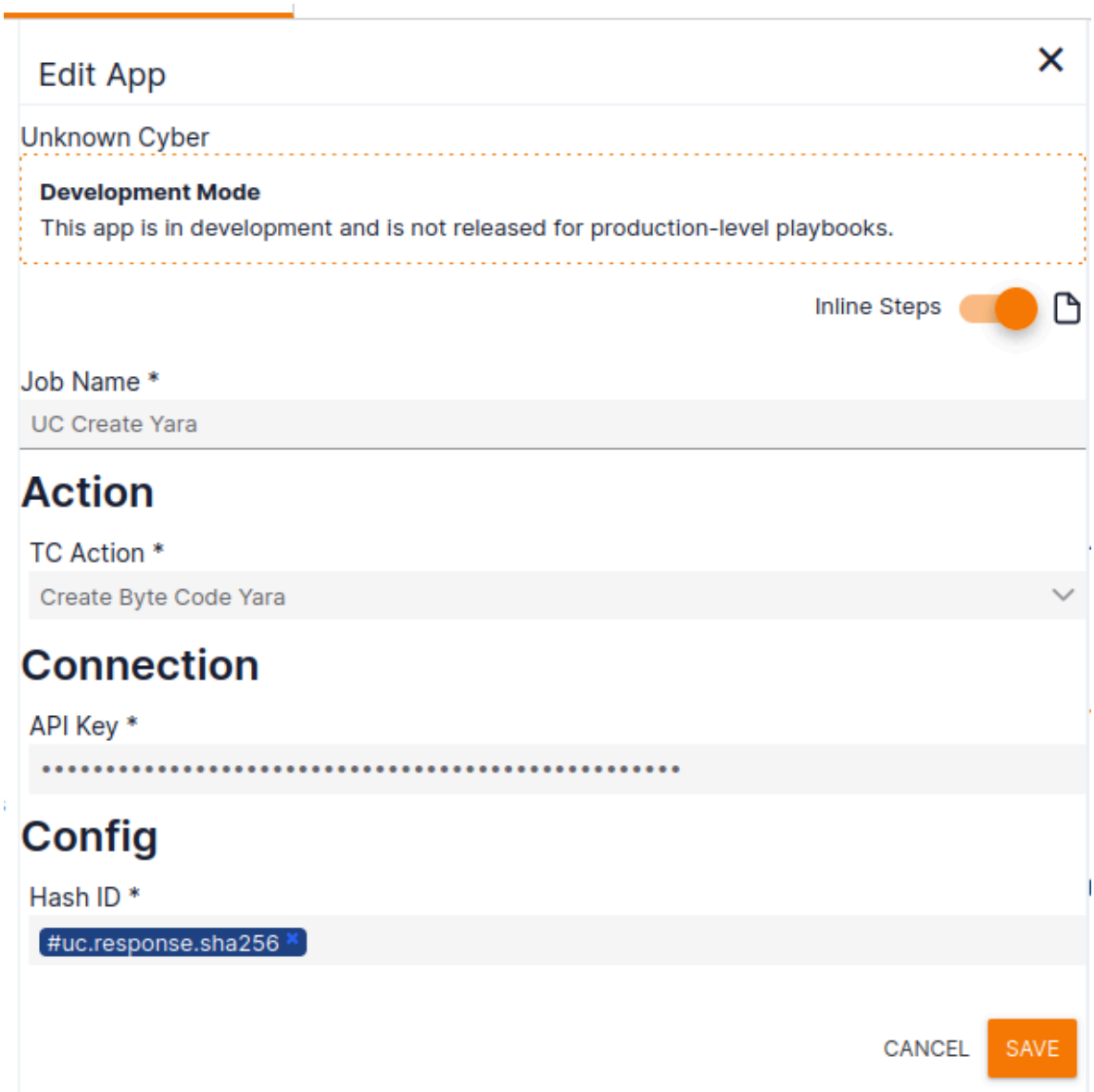
6. Following the **String Operator** app is an **If / Else** operator that checks the value of the String Operator. If the value is *false*, indicating the *match_list* contains 1.0 matches, then three apps run in parallel. The first app is a **Create ThreatConnect Attribute** which adds the match_list, a list of hashes, to the File Indicator as an attribute. The second app is another **Unknown Cyber** app that automatically creates a yara rule for the Hash of the File. The last app is another **ThreatConnect Notification** to alert to matches files. The configuration is below for these apps.



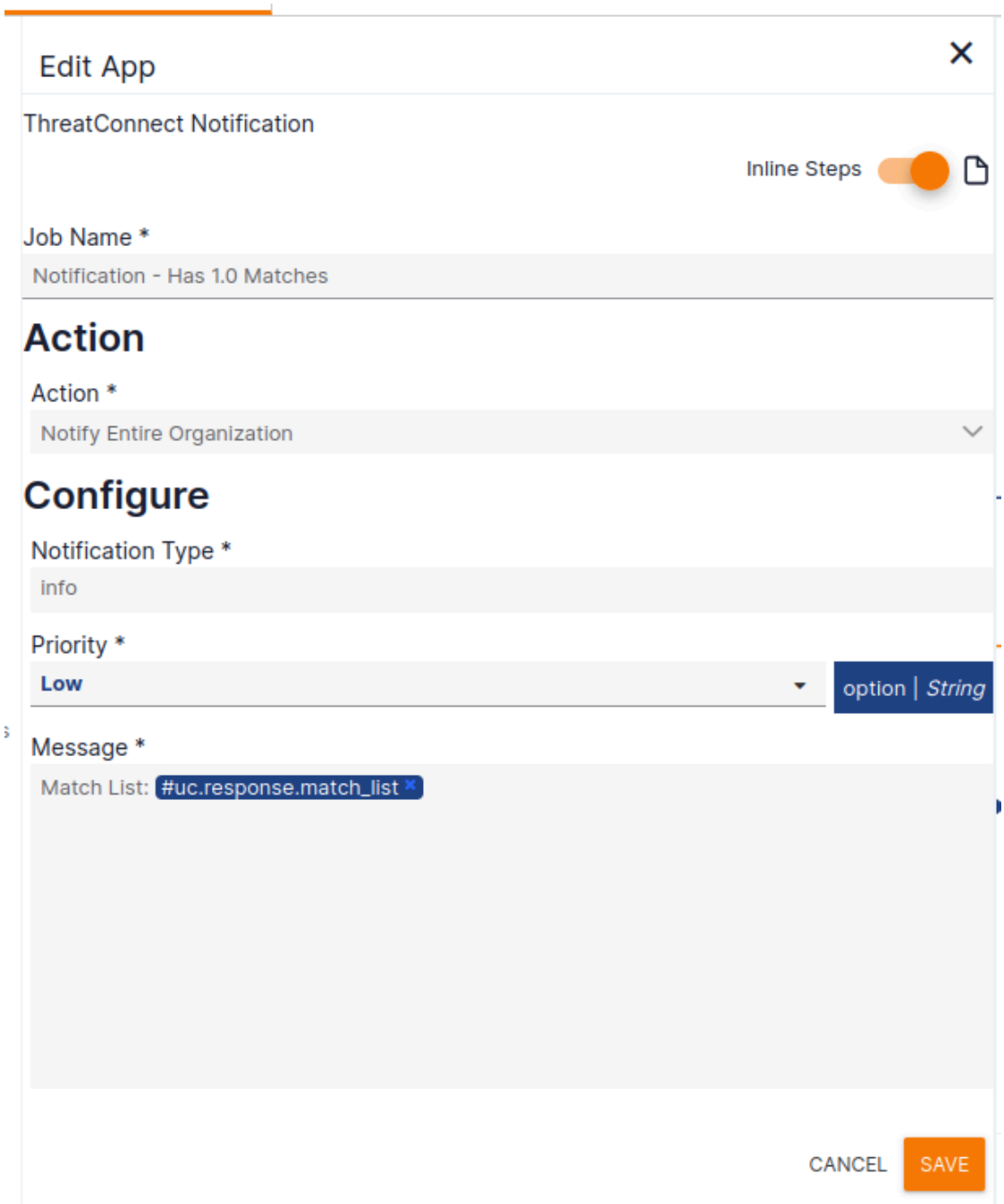
Create ThreatConnect Attribute	
Entity	#tc.file
Attribute Type	Additional Analysis and Context
Update if Exists	True
Attribute Content	1.0 matches: #uc.response.match_list



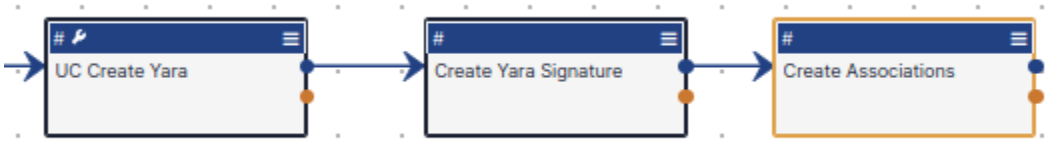
Unknown Cyber (Create Byte Code Yara)	
TC Action	Create Byte Code Yara
API Key	Your Unknown Cyber API Key
Hash ID	#uc.response.sha256



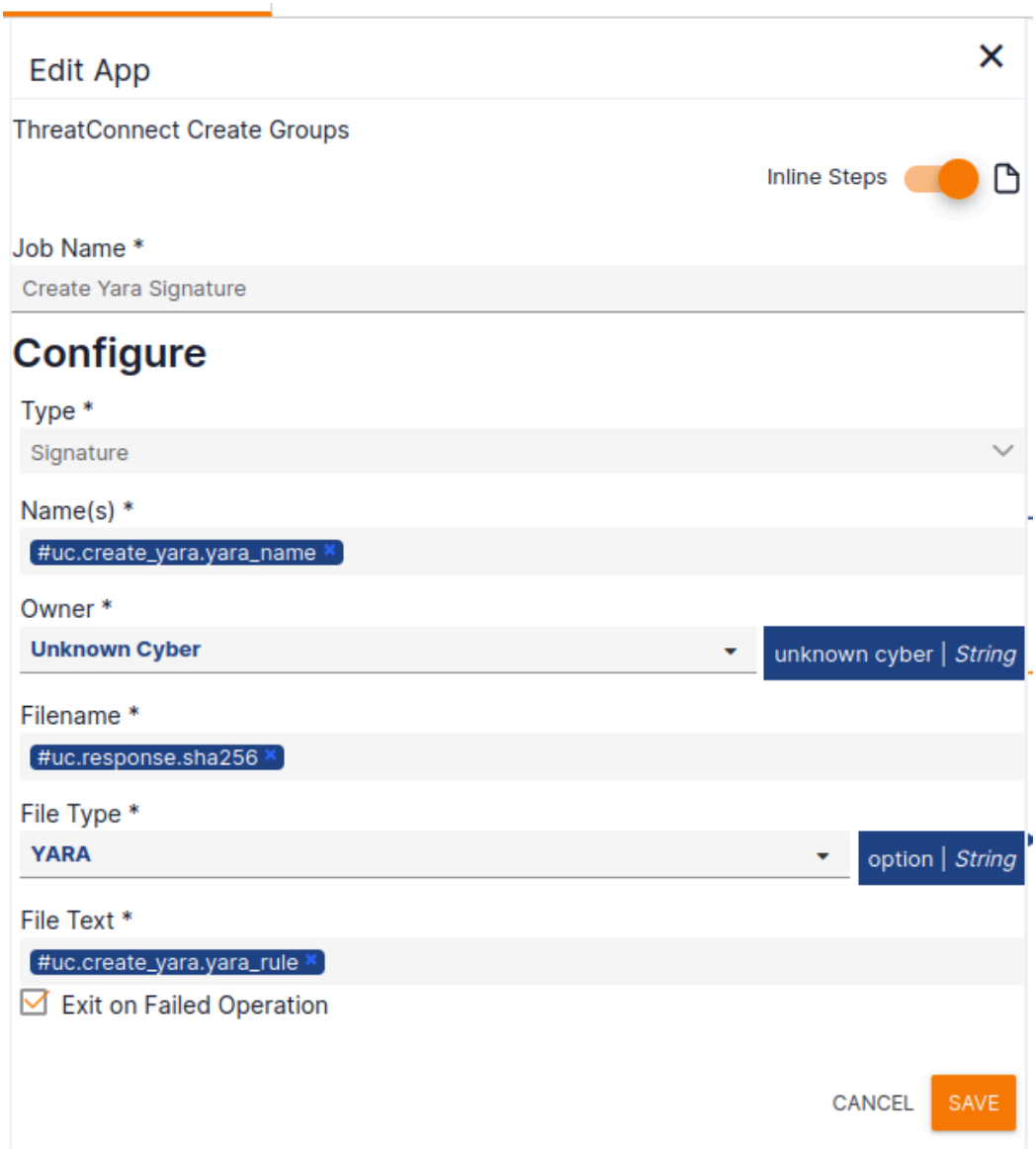
ThreatConnect Notification	
Action	Notify Entire Organization
Notification Type	Info
Priority	Low
Message	Match List: #uc.response.match_list



7. The last step of this playbook is to create a yara attribute and associate it to its File Indicator. The **ThreatConnect Create Groups** app is connected after **Unknown Cybers** Yara action to add the Yara signature in ThreatConnet. From there, a **ThreatConnect Associations** app is used to associate the Signature to the File Indicator.



ThreatConnect Create Group	
Type	Signature
Name(s)	#uc.create_yara.yara_name
Owner	Which Intel Source you want to trigger the playbook.
Filename	#uc.response.sha256
File Type	YARA
File Text	#uc.create_yara.yara_rule
Exit on Failed Operation	True



ThreatConnect Associations	
Action	Create Associations
Source(s)	#tc.file
Association(s)	#tc.groups
Apply to All	True
Exit on Failed Operation	True