



Spirion Extensions

**Spirion Automation: Box
Shield Classification
Labels**

Table of Contents

Spirion Automation: Box Shield Classification Labels.....	2
Introduction.....	2
Requirements.....	2
Important Notes.....	3
Software Versions.....	3
Box API.....	3
Process.....	3
Creating a Box Custom Application.....	3
Scripting the Box API.....	4
Configuring the Spirion Agent.....	5
Configuring Sensitive Data Platform (SDP).....	5
Uploading a Script.....	5
Creating a Playbook.....	5
Searching Box.....	6
Configuring Sensitive Data Manager (SDM).....	7
Creating a Workflow.....	7
Creating a Scan Policy.....	7
Running a Scan.....	8
Outcomes.....	9
Box Shield Labels.....	9
Spirion Scan Results.....	9
Troubleshooting.....	9
Spirion Scan Logs.....	9
API Script Logs.....	10
Manual Python Verification.....	10

Spirion Automation: Box Shield Classification Labels

Introduction

Accurate data discovery is “Step One” for proper data loss prevention (DLP), however not all solutions with DLP capabilities are equally equipped for the task.

This creates uneven footing as organizations attempt to implement privacy and security strategies across environments plagued by inconsistent visibility into fluctuating sensitive data footprints.

Spirion should instead serve as the single source of truth for software that lacks [Privacy-Grade data discovery](#). Rather than relying on a patchwork of search utilities built into individual platforms, information governance policies can be standardized around the precision of Spirion scan results – while still incorporating control mechanisms from external DLP tools that can be orchestrated either by [persistent metadata tags](#) or through API automation from Spirion’s scripting engine.

[Box Shield](#) is the DLP add-on to Box cloud storage. It includes classification labels with protective measures enforced through the application of said labels. Spirion enhances the searchability of sensitive data within Box, scanning for any conceivable combination of built-in [AnyFind™](#) and user-defined search terms, the results of which can then be scripted to programmatically apply classifications managed by Box Shield for ongoing protection.

Common support for Python across APIs, SDKs, and Spirion scripts make it ideal for prototyping how to integrate match results from a Spirion discovery scan with meaningful functionality from third-party applications. The following information serves as reference for automating scripted API interactions with Box Shield.

Released July 2023, © 2023 Spirion LLC.

Requirements

Before working on the steps outlined in this document, please confirm the following:

- The Spirion console (SDM or SDP) is up-to-date and accessible.
- A Spirion endpoint agent hosted on a Windows 10 system is online, polling to its console.

- Python is installed on the Windows host running the Spirion agent.
 - **NOTE:** LocalSystem must be able to run the Python script outlined in this document.
- A Box developer account is available.
- The same Box environment has Box Shield enabled.
- At least one Box Classification label is selected for testing.
 - Labels can be created from the Box Admin console.

Important Notes

Software Versions

SDP console version 22.Q3.1.226.1, SDM console version 11.8.2, Spirion agent version 12.5, and Python 3.11.1 were used to perform the steps described in this document.

Box API

The configuration explained below references the [Python implementation of Box's SDK](#) to create a script that applies Box Shield classification labels to Spirion search results via the [Box API](#).

Authentication is handled using [JSON Web Tokens \(JWT\)](#) authorized by a [Custom App](#) created in the Box Developer Console.

Process

Spirion can automate the execution of custom scripts with variables derived from scan results returned by Spirion agents. In SDM, this is handled from the [Action tab of any given Workflow](#), and from [Playbooks in SDP](#).

To apply Box Shield Classifications, Spirion is configured with a batch script that calls a Python script to interact with Box's API while passing in the Spirion match location (a Box file path) as a command-line argument using the "%Location%" variable. A sample Python script is available on the Spirion website in file box_shield.txt.

Creating a Box Custom Application

JWT authentication requires the creation of a Custom App in Box.

1. From the Box Dev Console, select **Create New App**.

2. Choose **Custom App**.
3. Ensure “Server Authentication (with JWT)” is selected and click **Create App**.
4. On the app’s **Configuration** tab, select the following under **Application Scopes**:
 - a. “Read all files and folders stored in Box” – should be selected by default
 - b. “Write all files and folders stored in Box”
 - c. “Manage users”
5. Enable “Make API calls using the as-user header” under **Advanced Features**.
6. Under **Add and Manage Public Keys**, choose to either **Add** or **Generate** the key(s) used for authentication.
 - a. This requires two-factor verification for the Dev console user editing the app.
7. **IMPORTANT:** If generating a new public/private keypair, Box automatically downloads a configuration JSON file – this will be needed later.
 - a. If using an existing public key, a similar file is available by clicking **Download as JSON** under **App Settings**.
8. Navigate to the **Authorization** tab and click **Review and Submit**.
9. Click **Submit** to finalize the Custom App.
10. Submitted applications must be approved from the Box Admin console by navigating to the **Custom Apps Manager** tab of the **Apps** menu.
 - a. **NOTE:** Any changes made to the Custom App after it has been submitted must be reapproved.

Scripting the Box API

Using Box’s Python SDK, the example script attached to this document’s Spirion Marketplace article executes the following:

1. Get the Spirion match result location – a file path terminating to a Box user account.
 - a. E.g. “example.address@email.com/box folder1/box folder2/file.docx”
2. Extract the following from this location:
 - a. Box account email
 - b. Breadcrumbs of the folder structure
 - i. This is used for match validation in step 4.
 - c. File name (and extension)
3. Box API Search for the file.

4. Validate that the file returned via API is the same file referenced in the Spirion match.
5. Box API Classification of file validated in the previous step.

NOTE: The Python sample file includes 3 variables that must reflect valid settings for the Box environment being used – `path_to_config`, `path_to_log`, and `shield_label` – defined in the “Customization” section (lines 10 through 17).

Configuring the Spirion Agent

In addition to moving the .JSON key file over to the Spirion agent, also install the Python script’s module dependencies for the Box SDK:

```
pip install --upgrade boxesdk "boxesdk[jwt]"
```

Configuring Sensitive Data Platform (SDP)

If using the SDP console, perform the following:

Uploading a Script

1. From the main menu, select **Settings**.
2. Navigate to **Script Repository**.
3. Click the **Actions** button toward the top-right of SDP and select “Add Script.”
4. Specify a **name** and **description** for the script.
5. Click the icon next to **Upload Script** and upload the following command as a batch file:

```
python c:\temp\box_shield.py %Location%
```

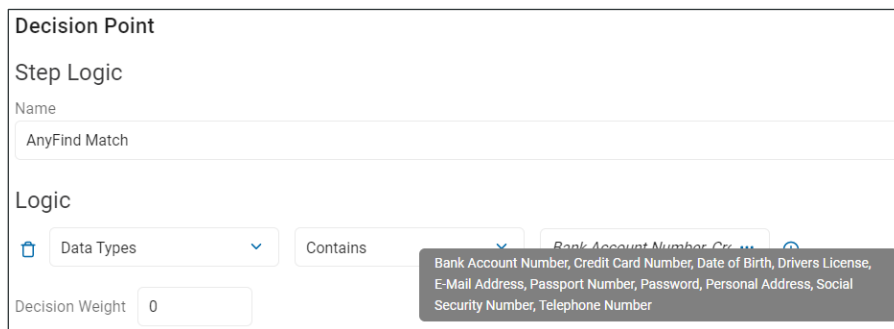
6. **Save** the uploaded script.

Creating a Playbook

The steps described below cover the relevant steps for automated script execution. For additional guidance, please refer to the [Spirion KB Article: How to Write a Playbook](#).

1. From the main menu, select **Scans**.
2. Navigate to **Scan Playbooks**.
3. Click **Add Playbook** using the button from the top-right.
4. Enter a **Name** and **Description** as desired.

5. Under the **Logic** section of a **Decision Point**, select the criteria that should trigger the rule.
 - a. For example, specifying “Data Types” and selecting any (or all) of the built-in AnyFind search parameters would trigger this script for every location (e.g. file/document) with 1 or more of the selected data types.



Decision Point

Step Logic

Name

AnyFind Match

Logic

Data Types Contains Bank Account Number, Credit Card Number, Date of Birth, Drivers License, E-Mail Address, Passport Number, Password, Personal Address, Social Security Number, Telephone Number

Decision Weight 0

6. From the **Select Action** pulldown menu, select “Execute Script” for the action card that corresponds to the intended decision path (yes/no) per the logic specified in the previous step.
7. Under **Select Script**, choose the script uploaded in the previous section.
8. Define additional Playbook logic as necessary before clicking the **Save** icon to finalize changes.

Searching Box

Once the Playbook has been created, it needs to be associated with a scan policy targeted at Box. Please reference the documentation under each step for additional guidance.

1. Add Box as a **Target** after navigating to **Data Assets and Targets** from the **Data Asset Inventory**.
 - a. [Spirion KB Article: Creating a Box Target – Authentication](#)
2. Create a new **Scan** by clicking the **Add Scan** button from the **Scans** screen.
3. Select the Playbook and Target configured per the guidance above.
 - a. [Spirion KB Article: Creating a Scan to Search Box](#)

Configuring Sensitive Data Manager (SDM)

If using the SDM console, perform the following:

Creating a Workflow

The steps described below cover the relevant steps for automated script execution. For additional guidance, please refer to the [Spirion KB Article: How to Write a Workflow Rule](#).

1. From the **Workflows** tab, select “Add” from the **Rule** pulldown menu.
2. Enter a **Name** and **Description** as desired.
3. On the **Definition** tab, select the criteria that should trigger the rule.
 - a. For example, specifying “Total Matches Greater Than or Equals 1” would trigger this workflow for every scan with 1 or more match results.



Definition:

X Total Matches Greater Than or Equals 1 +

4. On the **Endpoints** tab, select the agent(s) that will be searching Box for sensitive data.
5. On the **Actions** tab, enable **Perform the following remediation action** and select “Execute script.”
6. Click the ellipsis button (“...”) and upload the following command as a batch file:

```
python c:\temp\box_shield.py %Location%
```

7. Finalize the creation of this Workflow by clicking **Finish**.

Creating a Scan Policy

Once the Workflow has been assigned to the Spirion agent(s), Box is ready to be targeted as a repository in a discovery scan. Please reference the documentation under each step for additional guidance.

1. From the **Policies** tab, select “Create” from the **Policy** pulldown menu.
2. Configure the search’s data types by adjusting its **Settings**.
 - a. [Spirion KB Article: Getting Started with Scheduled Tasks](#)
3. Configure the search to target Box as a **Search Location**.
 - a. [Spirion KB Article: How to Search Box](#)

Running a Scan

As long as the Workflow and Scheduled Task have been applied to the same Spirion agent(s), running a console-initiated scan will trigger the scripted interactions with Box's API.

1. From the **Policies** tab, select the Scheduled Task Policy from the **Policy List**.
2. Scans can be initiated in one of two ways:
 - a. One-off scan:
 - i. Right click the policy and select **Initiate Search** from the **Search** menu.
 - b. Scheduled scan:
 - i. Navigate to **Scheduled Tasks** and click **Add** from the ribbon menu.
3. The scan will initiate, searching the Box account(s) specified in the policy created per the previous section.

Outcomes

The accuracy and flexibility of Spirion's built-in and custom search capabilities combined with the protective measures enabled by Box Shield automation is a seamless integration with many benefits:

1. Scan for sensitive data types not included with Box Shield.
2. Define custom search terms using [pattern matching via regular expressions](#) or even Spirion's [algorithmically validated Search API](#).
3. Create compound search terms using Spirion's [Sensitive Data Definitions](#).
4. Standardize the discovery process across various target repositories by using Spirion to scan other unstructured, structured, or cloud data targets.
5. Centralize scan results to an enterprise console that reports on results across the entire sensitive data footprint.

Box Shield Labels

The automated script runs at the end of a Spirion scan, after all locations have been evaluated for matches. Box Shield labels will be applied automatically at this time.

Spirion Scan Results

Once completed, scan results are returned to the Spirion console. Match locations will include the associated Box user's email address as the root path for each file reported.

Troubleshooting

Spirion Scan Logs

Custom script execution can be verified by auditing Spirion scan logs. In addition to confirming a successful connection to the repository environment targeted in a scan, logs will also indicate that the workflow script has been executed for any given match location:

```
"USER ACTION" "Successfully executed script (via workflow) on the file..."
```

However, the statement above only indicates that the batch file ran. Testing the Python script manually is necessary if the expected outcome of the Python script is not observed

API Script Logs

Similarly, the Python script used in this article writes logs to a user-defined location or `C:\temp` by default. Logs are configured to rotate hourly, with success and failure messages confirming the outcome of automated API requests informed by Spirion scan results.

NOTE: Optional logging can be enabled for additional verbosity by leaving *line 200* uncommented in the example script.

Manual Python Verification

Console-initiated scans control the Spirion (Windows) agent using the LocalSystem account by default. Since Spirion search logs only indicate if the batch file ran properly, any Python errors will not be reported.