A blue text on a black background

Description automatically generated

Amazon Security Lake

IdentityNow Integration Guide

Version 1.0

Eric Panico, Dhara Shah

Table of Contents

[Integration Overview 3](#_Toc146623977)

[Integration Setup 4](#_Toc146623978)

[Step 1: Custom Sources Setup 4](#_Toc146623979)

[Step 2: IdentityNow OAuth2.0 Configuration 5](#_Toc146623980)

[Step 3: AWS CloudFormation 6](#_Toc146623981)

[CloudFormation Setup 8](#_Toc146623982)

[CloudFormation Parameters 10](#_Toc146623983)

[Testing Your Integration 11](#_Toc146623984)

[Schedule the Lambda Function 15](#_Toc146623985)

# Integration Overview

This integration is designed to extract audit events from an IdentityNow tenant, transform them into an OCSF schema, and save them to a custom source location in Amazon Security Lake in Apache parquet format. The components of this integration are:

**Lambda Package**: sailpoint\_ocsf\_lambda\_package.zip contains a python script, libraries, and json files for processing audit events from the IdentityNow API.

**Glue ETL script:** sailpointOCSFGlueETL.py is a Python script that transforms the json output provided by the Lambda function into a parquet format for storage in Amazon Security Lake. This is provided in a separate script because Lambda does not support the Spark libraries needed for creating parquet files.

**Cloud Formation script**: cloudformation.yaml is a cloud formation template that creates a resource stack that includes most of the resources needed to use this integration.

Prior to beginning the setup process for this integration an Amazon Security Lake account must be enabled and custom sources must be setup. For an overview please see the [Amazon Security Lake Getting Started guide](https://docs.aws.amazon.com/security-lake/latest/userguide/getting-started.html) and [Collecting data from custom sources](https://docs.aws.amazon.com/security-lake/latest/userguide/custom-sources.html).

You will need to create three custom sources for collecting OCSF data from SailPoint IdentityNow. Each source will be customized to handle the OCSF schema appropriate for the IdentityNow event types transformed by the integration.

# Integration Setup

## Step 1: Custom Sources Setup

1. Open the [Amazon Security Lake console](https://console.aws.amazon.com/securitylake/)
2. Click on Custom Sources in the left navigation menu.
3. Click the Create custom source button.
4. In Custom source details enter the following information:

* Data source name: sailpoint-auth \*
* OCSF Event Class: Authentication
* AWS account ID: The AWS Account ID where the SailPoint integration resource stack will reside (this can be the same account that Security Lake resides or any other AWS account)
* External ID: Must be “SailPoint\_” followed by the tenant name associated with your IdentityNow Tenant (https://{SailPointTenantName}.identitynow.com)
* Service Access: Choose “Create and use a new service role”

1. Click the Create button.
2. Repeat this process 2 more times for the following Data Source Names and Event Classes:

* Data source name: sailpoint-acct-chng\*
* OCSF Event Class: Account change
* Data source name: sailpoint-sched-job\*
* OCSF Event Class: Scheduled Job Activity

\*It is important that these data source names are input exactly as provided for the integration to function properly.

## Step 2: IdentityNow OAuth2.0 Configuration

|  |  |
| --- | --- |
| You will need to create a personal access token in IdentityNow which can then be used to generate access tokens to authenticate API calls. [See this guide](https://developer.sailpoint.com/idn/api/authentication/#generate-a-personal-access-token) for instructions on how to generate a personal access token.  **Note:** When you setup your personal access token, be sure to enable the scope ‘sp:search:read’ and *only this scope*.  You will need your Client ID and Client Secret during the Cloud Formation process. |  |

## Step 3: AWS CloudFormation

AWS Cloud Formation allows the provisioning of a resource stack with code. Your integration includes cloudformation.yaml which is a template specifically designed to provision all of the resource components necessary to enable the integration. We recommend using this template to provision AWS resources. The template provisions the following resources:

|  |  |  |
| --- | --- | --- |
| Logical ID | Type | Purpose |
| CopyFilesToS3 | AWS::Lambda::Function | Lambda function to copy source code from GitHub to S3. |
| JsonClassifier | AWS::Glue::Classifier | Used by SailPointGlueCrawler to collect metadata from json audit event data. |
| LakeFormationDatabaseCrawlersPermissions | AWS::LakeFormation::Permissions | Grants permission for Glue to use a database to store and access metadata collected by a Crawler . |
| LakeFormationDataLocationCrawlerPermissions | AWS::LakeFormation::Permissions | Grants permission for a Glue Crawler to collect metadata from json audit event data. |
| LakeFormationDataLocationResource | AWS::LakeFormation::Resource | Gives Lake Formation permission to use a service linked role to access json log data. |
| OCSFDatabase | AWS::Glue::Database | Database to store audit event data temporarily before it is transformed from json to parquet. |
| S3Copy | Custom::CopyToS3 | Custom code used by CopyFilesToS3 Lambda function. |
| S3CopyLambdaExecutionRole | AWS::IAM::Role | Gives CopyFilesToS3 Lambda function permission to write files to SourceCodeS3Bucket. |
| SailPointGlueCrawler | AWS::Glue::Crawler | Collects metadata from json audit event data, needed for parquet file conversion. |
| SailPointGlueETLJob | AWS::Glue::Job | Runs the code in sailpointOCSFGlueETL.py to transform data from json to parquet. |
| SailPointLambdaAssumableRole | AWS::IAM::Role | Grants permission for SailPointLambdaFunction to assume roles created during custom source creation. |
| SailPointLambdaExecutionRole | AWS::IAM::Role | Grants various permissions for SailPointLambdaFunction on SailPointOCSFGlueCrawler, OCSFDatabase, SailPointSSMParameter and TempFileS3Bucket |
| SailPointLambdaFunction | AWS::Lambda::Function | This is the main function that will execute the data ETL process. |
| SailPointOCSFGlueRole | AWS::IAM::Role | Grants permission for Glue resources on TempFileS3Bucket, SourceCodeS3Bucket and Lake Formation |
| SailPointSSMParameter | AWS::SSM::Parameter | This parameter is used to store the date/time of the last event captured from the IdentityNow API. |
| SecretManagerResource | AWS::SecretsManager::Secret | This resource is used to store the IdentityNow personal access token Client Secret value. |
| SourceCodeS3Bucket | AWS::S3::Bucket | A bucket used to store code used by SailPointGlueETLJob and SailPointLambdaFunction |
| TempFileS3Bucket | AWS::S3::Bucket | Temporary storage for transformed audit events in json format |

### CloudFormation Setup

1. Open the [CloudFormation console](https://console.aws.amazon.com/cloudformation/).
2. In the top right corner click the **Create Stack** button and choose **With new resources (standard).**

A screenshot of a computer

Description automatically generated

1. In the Specify template section choose **Upload a template file.**

A screenshot of a computer

Description automatically generated

1. Click the **Choose file** button and browse to the cloudformation.yaml file provided, then click Next.
2. In the Stack name section choose a name for this resource stack (For example, SailPointOCSFResourceStack).
3. Review the Parameters section carefully. Some values have been provided as default values that you can change to fit your needs. Values are required for each parameter. When completed, click **Next.**

### CloudFormation Parameters

**AccountChangeExecutionRole** - **Provider Role ARN** associated with the Amazon Security Lake custom Source name sailpoint-acct-chng. This should be copied from your Security Lake console.

* Open your Security Lake console.
* Select **Custom sources** from the navigation menu
* Copy the value in the **Provider role ARN** column associated with the Custom source name *sailpoint-acct-chng*

**AuthenticationExecutionRole** - **Provider Role ARN** associated with the Amazon Security Lake custom Source name sailpoint-auth. This should be copied from your Security Lake console.

* Open your Security Lake console
* Select **Custom sources** from the navigation menu
* Copy the value in the **Provider role ARN** column associated with the Custom source name *sailpoint-auth*

**MaxEventsFromAPI -** Number of events to request from SailPoint API on each run (10000 max)

**SailPointClientID** - The ClientID from the Personal Access Token you created in your IdentityNow Tenant ([from Step 2](#_Step_2:_IdentityNow))

**SailPointClientSecret** - The Client Secret from the Personal Access Token you created in your IdentityNow Tenant ([from Step 2](#_Step_2:_IdentityNow))

**SailPointTenantName** - The tenant name associated with your IdentityNow Tenant (https://{SailPointTenantName}.identitynow.com)

**ScheduledJobActivityExecutionRole** - **Provider Role ARN** associated with the Amazon Security Lake custom Source name sailpoint-sched-job. This should be copied from your Security Lake console.

* Open your Security Lake console.
* Select **Custom sources** from the navigation menu.
* Copy the value in the **Provider role ARN** column associated with the custom source name *sailpoint-sched-job.*

**SecurityLakeAccountID** - Enter the AWS Account ID associated with your Amazon Security Lake configuration. This is the account that has your custom sources configured.

**SecurityLakeDatabaseName** - The name of the database for Amazon Security Lake data. This value can be found in your AWS Lake Formation console.

* Open your Security Lake console.
* Select **Summary** from the navigation menu.
* Select **Lake formation database** from the Quick links section.
* Copy the database name from the Database details section.

**SecurityLakeExternalID** - The External ID associated with each Amazon Security Lake custom source. This should be set to same value used for External ID when you setup your custom sources.

**SecurityLakeS3Bucket** - Name of the S3 bucket where your Amazon Security Lakes Custom Source data is stored. This should be copied from your Security Lake console.

* Open your Security Lake console.
* Select **Regions** from the navigation menu.
* Click on the S3 location associated with the region where your custom source is hosted.
* Copy the bucket name at the top of the page (should start with “aws-security-data-lake”)

# Testing Your Integration

1. From the CloudFormation stacks screen click on the name of your stack and select the Resources tab.
2. Click the link to SailPointOCSFLambdaFunction in the Physical ID column.

A screenshot of a computer

Description automatically generated

1. Click on the Test tab, then click on the Test button. You will see the Executing Function message in the information dialog box. Depending on the value you specified for **MaxEventsFromAPI** the operation will take from 5 to 10 minutes to complete. You can click on Details to see a summary of the operation after completion.

A screenshot of a test

Description automatically generated

1. Open your S3 console and browse to the location for your Security Lake custom source data. You should see one folder for each of the OCSF custom sources you created. Within each folder you should see the following hierarchy representing the partitioned data: ***region=region/accountId=accountID/eventDay=YYYYMMDD/***

A screenshot of a computer

Description automatically generated

1. Check one of the eventDay folders for the parquet file data left by the script. Select the checkbox next to one parquet file, then select **Query with S3 select** from the Actions menu.

A screenshot of a computer

Description automatically generated

1. In the Input settings section make sure Apache Parquet is selected.

A screenshot of a computer

Description automatically generated

1. In the Output settings section select JSON Format.

A screenshot of a computer

Description automatically generated

1. In the SQL query section, use the default query to select 5 records from the file.

A screenshot of a computer

Description automatically generated

1. Click the Run SQL query button. You should see 5 events converted to the OCSF schema\*

\* You should notice that the first object in each record named “data” contains the original source record from the SailPoint API. “data” is an optional JSON object provided in the [OCSF v1.0.0-rc.2 schema](https://schema.ocsf.io/1.0.0-rc.2/) that has been chosen to use to store the original event data and schema for reference.

# Schedule the Lambda Function

We recommend running the Lambda function at intervals no more frequent than once every hour. You can use [Amazon EventBridge](https://aws.amazon.com/eventbridge/) to accomplish this. See [Tutorial: Schedule AWS Lambda functions using EventBridge](https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-run-lambda-schedule.html)