CASB Connect API Requirements

CASB Connect allows for seamless integration of cloud service APIs to enforce security controls such as Data Loss Prevention, Secure Collaboration, and Activity Monitoring.

CASB Connect relies on the APIs provided by a cloud service in order to enforce the following use cases.

- Enforce Data Loss Prevention (DLP) policies to protect sensitive and regulated data.
- Enforce internal and external collaboration policies.
- Inspect user and administrator activity within the service.
- Detect internal and external threats using Skyhigh Security's machine learning algorithms.

How It Works

Skyhigh Security relies on **User Activity/Event feed** from the CSP with necessary information regarding each activity to provide its feature set. An integration between Skyhigh and the service gives the necessary grants and configurations to Skyhigh Security for retrieving this information from the service. Integrating a service with Skyhigh involves three major components.

1. **Authorization**. Skyhigh must be authorized with the service to receive the user activity feed and to retrieve necessary information for each event. When a Security Administrator enables the API access for a cloud service from Skyhigh, Skyhigh uses OAuth2.0 authorization flow for getting an access token from the service. This access token is used later to call required APIs. For details on OAuth2.0, see https://oauth.net/2/.
2. **Activity Monitoring and DLP**. For every user activity performed in a cloud service, Skyhigh expects to receive an event/activity (in JSON) representing that user activity. Skyhigh can receive a service's user activity feed either through a Pull or Push model.
    a. **Pull Model** – In the Pull model, Skyhigh polls the cloud service's event APIs periodically to get a list of events that occurred since the last polling (**checkpoint**). When a Security Administrator disables API access for a service, Skyhigh will stop polling for events.
    b. **Push Model (preferred)** – The Push model facilitates a serivce to push activities to Skyhigh as and when they occur using webhooks. When API access is enabled for a service, Skyhigh will register the webhook with the service for the event feed. The webhook is registered by making an API call to the service. When a Security Administrator disables API access for the service, Skyhigh will deregister webhooks via API with the service to stop receiving the event feed.
3. **Response Actions.** If a policy violation is detected during the evaluation of DLP policies defined within Skyhigh, Skyhigh will perform response actions as specified in the corresponding policy. Generic response actions such as Policy Incident creation and Email Notification to a specific email are available by default from Skyhigh. Service-dependent response actions such as Delete (deleting the content that caused the DLP violation), Quarantine (moving sensitive content to a temporary location where an admin can review and choose to restore or delete),

Modify sharing permissions (when sensitive data is shared with external users) or others, require Skyhigh to make additional API calls to perform these actions. Service-dependent response actions are supported only if the service provides all the necessary APIs to perform those actions.

Prerequisites

1. The service must support OAuth 2.0 with an Authorization Code grant to allow Skyhigh Security to make Rest API calls with a bearer access taken. The service must support the global OAuth 2.0 client, which can be created and granted access to any cloud service tenant.
2. There must be at least one user role that can grant Skyhigh permissions to monitor activity and access content for all users. In other words, access tokens acquired by Skyhigh through OAuth 2.0 should work for accessing any API reports and content in the service.
3. **Pull Model.** For the pull model, the service must have a REST API that provides Skyhigh with a log of all user activities being performed in the service with eventual consistency.
4. **Push Model.** For the push model, the service must have webhooks support, which delivers user activity payloads to a registered Skyhigh endpoint. Webhooks must support registration at the enterprise-tenant level to retrieve activities from across the tenant. It should require Skyhigh to register only once for a webhook to retrieve all tenant activity, including the activity of new objects created/added later to the service tenant. The service must provide an API for registering and deregistering a webhook. Once a webhook is registered via API, the service must send all user activity in JSON format to the specified endpoint.
5. **Activity/Event JSON:** All activities must have the following attributes as part of the JSON object:
   a. **Tenant ID –** Unique identifier of the cloud service tenant.
   b. **Event ID –** Unique identifier of the user activity.
   c. **Event Name –** Name of the user activity performed (for example, File Created, File Updated, etc.).
   d. **Event Timestamp –** Date and time the user activity is performed.
   e. **Log Timestamp –** Date and time the performed user activity is logged. This is a must when events are not logged in an eventually consistent manner.
   f. **Event Type –** Type of activity performed. (for example, Data Creation, Data Deletion, Administration, etc.).
   g. **Actor ID –** Unique identifier of the user who performed the activity.
   h. **Actor Name –** Full Name of the user who performed the activity.
   i. **Actor Email –** Email ID of the user who performed the activity.
   j. **Actor Type –** Indicates if the user who performed the activity is Internal/External to the organization.
   k. **Actor Role –** Role of the user who performed the activity.
   l. **Actor IP Address –** IP Address of the user who performed the activity.
   m. **Object ID –** Unique identifier of the object (for example, File, Message, Note, Ticket, Case, etc.) on which the activity was performed.
   n. **File Download URL –** Download URL for the object/file on which activity is performed. The content of the object must be downloadable with this URL with the access token acquired.
   o. **Object Type –** Type of the object on which activity has been performed (for example, File, Folder, Case, Message, Post, etc.).
   p. **Object Name –** Name of the object on which activity has been performed.
   q. **Object Path –** Location of the object on which activity has been performed.

r. **From Email –** Email of the user who initiated collaboration/ sharing of the object.
s. **Target User Email –** Email of the user with whom the object is collaborated/shared.
t. **Target User ID –** Unique Identifier of the user with whom the object is collaborated/shared.
u. **Target User Type –** Indicates if the target user of the collaboration is internal/external to the organization.
v. **Role –** Role assigned to target user in the collaboration/ sharing (for example, Editor, Owner, Viewer, etc.).
w. **Shared Link –** Sharing Links active/created on the object.
x. **Visibility –** Visibility of the shared link (for example, Public, Organization, Restricted, etc.).

NOTE: If the event JSON does not contain any of the above information, the service must at least have APIs to fetch this additional information based on the metadata returned in events.