Prisma Cloud Continuous Permission Management

Limited GA

PCS 24.2.2

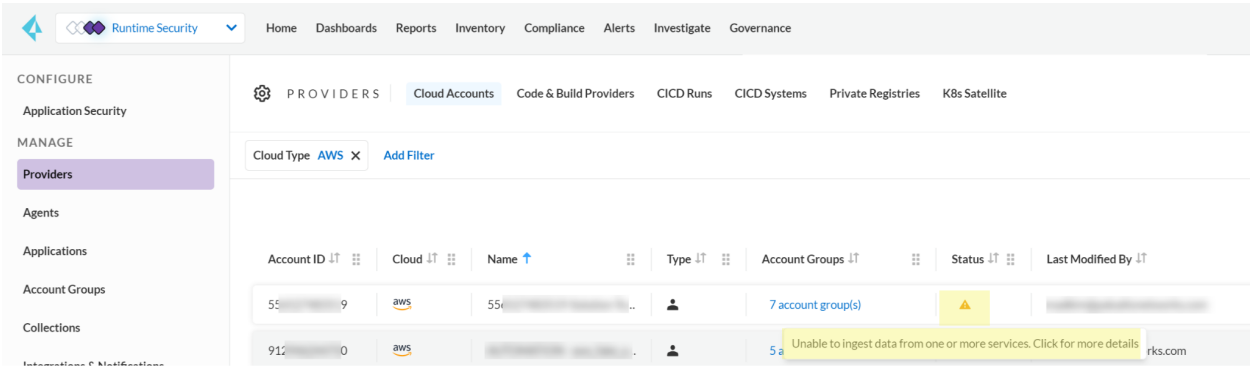
408.753.4000 | 3000 Tannery Way, Santa Clara, CA 95054 | paloaltonetworks.com

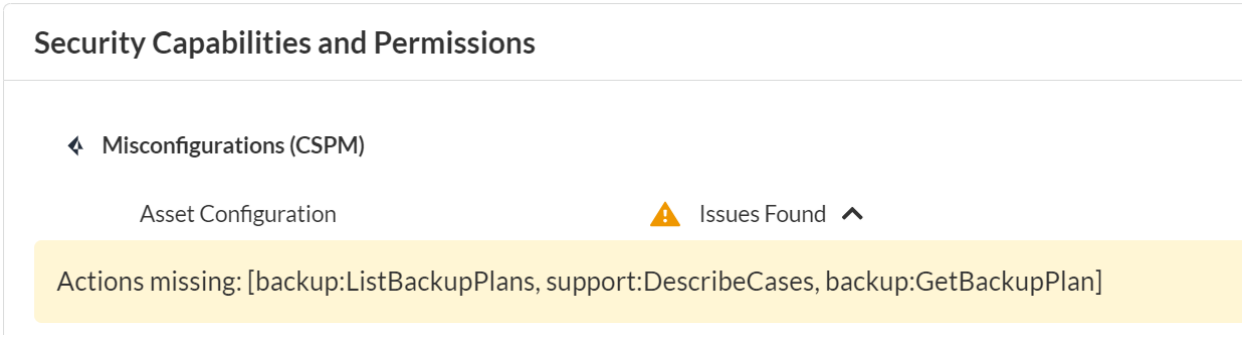
Table of Contents

Overview

Prisma® Cloud provides an enhancement designed to streamline your experience with your onboarded cloud accounts. With this enhancement, you can now easily manage the required API ingestion permissions for your cloud accounts continuously, ensuring smooth operation and eliminating potential misconfiguration issues.

Upon reviewing your onboarded cloud account status under **Settings > Providers > Cloud Accounts**, you may encounter a **warning** status indicating ***Issues found***, often due to missing permissions required for ingesting APIs.





To address this, Prisma Cloud provides a convenient solution: a Python script that automates granting necessary permissions. With this script, you can now save time and effort by proactively addressing and future-proofing missing permission issues against new API ingestions.

Getting Started

Assumptions

1. You have Administrator permissions in the Cloud Service Provider (AWS, Azure, GCP, and OCI) account to run the CFT (Cloud Formation Template) or TFT (Terraform Template).
2. Admin Access to Prisma Cloud Console.
3. You have already onboarded your AWS/Azure/GCP/OCI accounts onto Prisma Cloud.
4. Be comfortable granting the **read-all** type of permissions to Prisma Cloud.
5. Have a working knowledge of how to use Prisma Cloud and its features.
6. You have Python versions 3. x.x and higher installed on your system.

How to Enable AutoConsent Script?

Perform the following steps to enable the **AutoConsent** Python script:

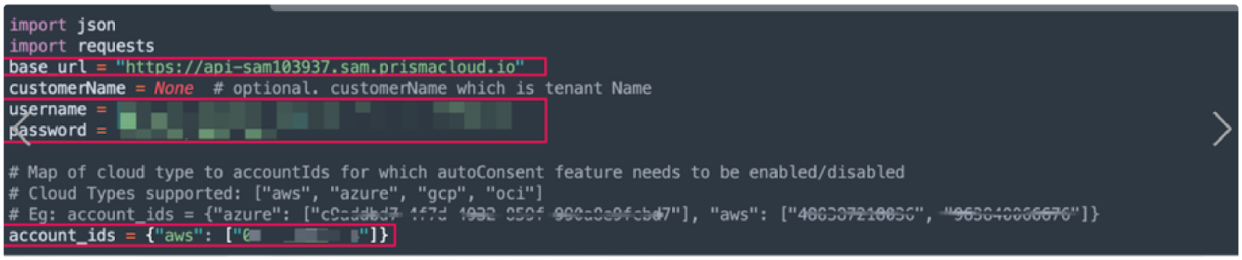
You can enable auto permissions for both standalone or organization accounts. However, you will not be able to enable or disable permissions individually for member accounts within the organization account of your Cloud Service Provider (CSP).

1. Copy the following Python script and enter the required details:

    1. Base URL: [Provide the URL]. For example, *https://app2.prismacloud.io/login*
    2. Prisma Admin Username and Password: [Enter Credentials]
    3. CSV values of cloud types and the account ID: [Enter CSV Values]

        For example:

        {"azure": ["c9addbd7-4f7d-4932-859f-990a8e9fcbd7"], "aws": ["488387218036", "963840066676"]}]

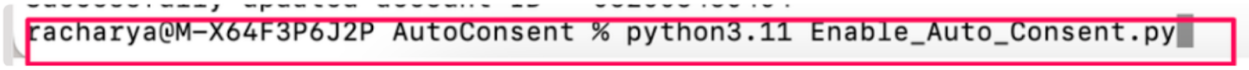


Python

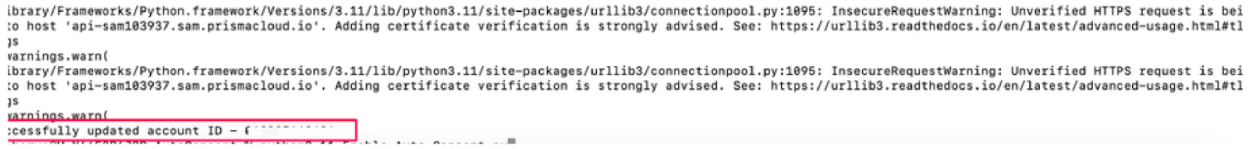
```python
import json
import requests
base_url = "https://api-sam103937.sam.prismacloud.io"
customerName = None # optional. customerName which is tenant Name
username = "your prisma admin username"
password = "password"

# Map of cloud type to accountIds for which autoConsent feature needs to be
enabled/disabled
# Cloud Types supported: ["aws", "azure", "gcp", "oci"]
# Eg: account_ids = {"azure": ["c9addbd7-4f7d-4932-859f-990a8e9fcbd7"], "aws":
["488387218036", "963840066676"]}
account_ids = {"aws": ["0123456789"]}

# autoConsent = "enabled" - to opt in accountIds for autoConsent
# autoConsent = "disabled" - to opt out accountIds for autoConsent
auto_consent_value = "enabled"

# Prerequisite: Obtain an authorization token by Logging In.
login_url = f"{base_url}/login"

if customerName is not None:
  login_payload = json.dumps({
   "customerName": customerName,
   "username": username,
   "password": password
  })
else:
  login_payload = json.dumps({
    "username": username,
    "password": password
  })
```

```
login_headers = {
 'Content-Type': 'application/json'
}

response = requests.request("POST", login_url, headers=login_headers,
data=login_payload, verify=False)
YOUR_TOKEN = response.json()['token']

# Call Cloud Account Patch API to enable/disable autoConsent
cloud_account_patch_api_url = "{base_url}/cloud/{cloudType}/{accountId}"

cloud_account_patch_api_payload = json.dumps({
 "autoConsent": auto_consent_value
})

cloud_account_patch_api_headers = {
 'accept': 'application/json',
 'content-type': 'application/json',
 'x-redlock-auth': YOUR_TOKEN
}

for cloud_type, account_ids in account_ids.items():
  for account_id in account_ids:
    response = requests.request(
      method="PATCH",
      url=cloud_account_patch_api_url.format(base_url=base_url,
cloudType=cloud_type, accountId=account_id),
      headers=cloud_account_patch_api_headers,
      data=cloud_account_patch_api_payload,
      verify=False
    )

    if response.status_code == 200:
      print(f"Successfully updated account ID - {account_id}")
    else:
      print(f"Failed to update account ID {account_id}, status_code:
{response.status_code}")
      print(f"Failed to update account ID {account_id}, response:
{response.text}")
      print(f"Failed to update account ID {account_id}, status:
{response.headers.get('x-redlock-status')}")
```

4. Save and run the script.

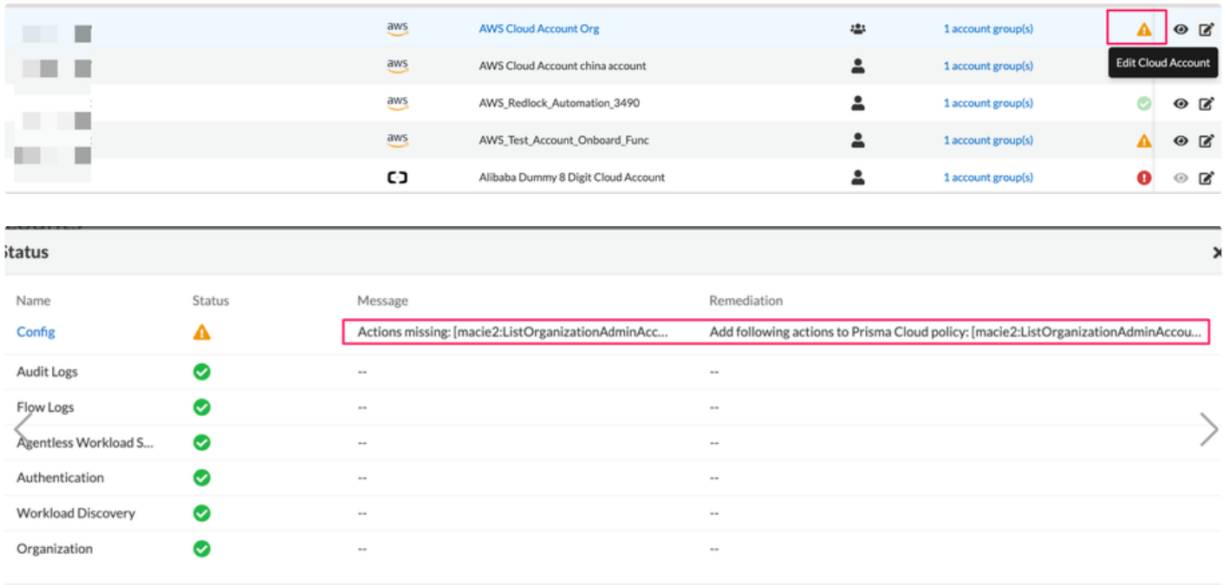


You will receive a success message after successfully executing the script.



2. Log in to Prisma Cloud. Go to the **Settings > Providers > Cloud accounts** page to modify the desired account.

Note: The **Status** will still show a **Warning,** it will turn **Green** after you update the CFT/TFT template.



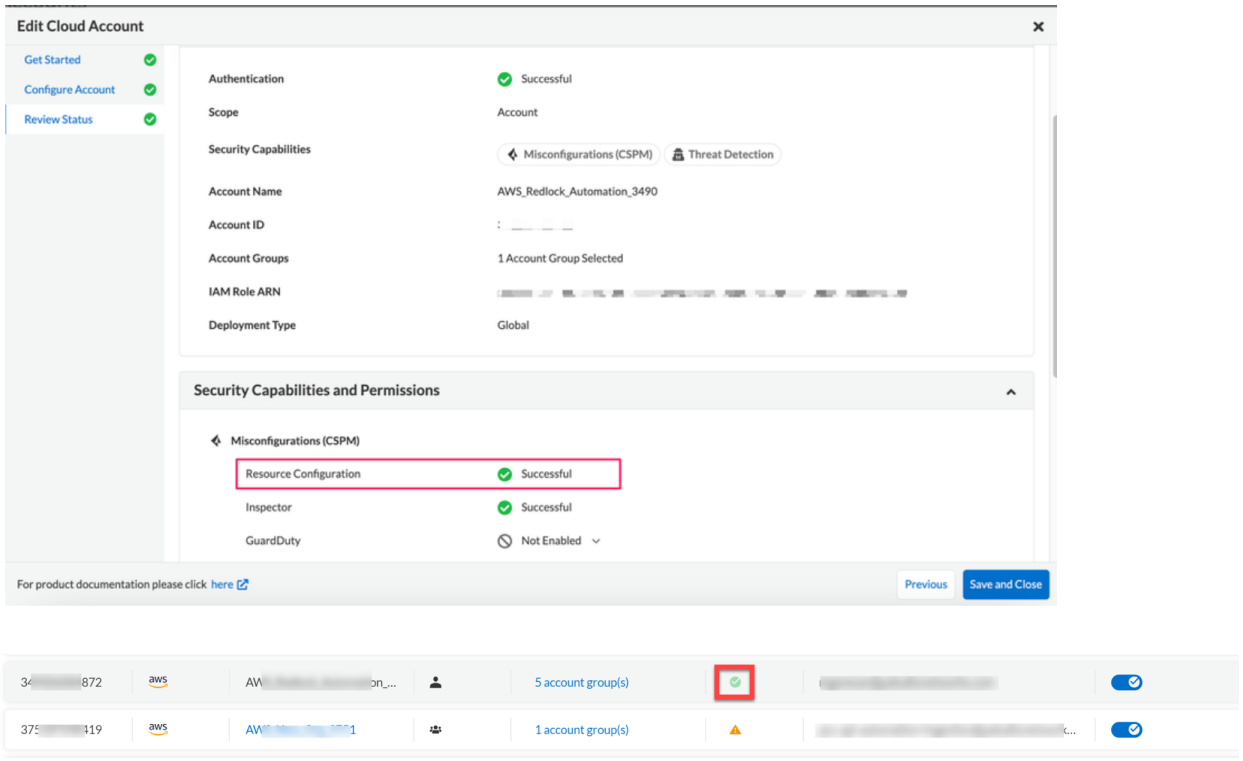
3. Perform the following steps as listed to update the CFT/TFT template for the required cloud account:

- Update an Onboarded AWS Organization
- Update an Onboarded AWS Account
- Update Azure Application Permissions

- Update an Onboarded GCP Account
- Update Permissions on an Onboarded OCI Account

4. Once the account is successfully updated, go to the **Cloud Account** page and notice that the account **Status** must have turned green.

Note: The status will remain green even when new APIs are automatically ingested on Prisma Cloud.



Note: You may need to run CFT or TFT templates when certain new APIs require explicit permissions that are outside the scope of roles assumed by Prisma Cloud. This is typically the CSP IAM implementation.

How to Disable AutoConsent Script?

To disable the script, copy the following script and perform all the steps listed in How to Enable AutoConsent Script.

Python

```python
import json
import requests
base_url = "https://api-sam103937.sam.prismacloud.io"
customerName = None # optional. customerName which is tenant Name
username = "your prisma  admin username"
password = "password"

# Map of cloud type to accountIds for which autoConsent feature needs to be
enabled/disabled
# Cloud Types supported: ["aws", "azure", "gcp", "oci"]
# Eg: account_ids = {"azure": ["c9addbd7-4f7d-4932-859f-990a8e9fcbd7"], "aws":
["488387218036", "963840066676"]}
account_ids = {"aws": ["0123456789"]}

# autoConsent = "enabled" - to opt in accountIds for autoConsent
# autoConsent = "disabled" - to opt out accountIds for autoConsent
auto_consent_value = "disabled"

# Prerequisite: Obtain an authorization token by Logging In.
login_url = f"{base_url}/login"

if customerName is not None:
  login_payload = json.dumps({
   "customerName": customerName,
   "username": username,
   "password": password
  })
else:
  login_payload = json.dumps({
    "username": username,
    "password": password
  })

login_headers = {
 'Content-Type': 'application/json'
}
```

```
response = requests.request("POST", login_url, headers=login_headers,
data=login_payload, verify=False)
YOUR_TOKEN = response.json()['token']

# Call Cloud Account Patch API to enable/disable autoConsent
cloud_account_patch_api_url = "{base_url}/cloud/{cloudType}/{accountId}"

cloud_account_patch_api_payload = json.dumps({
 "autoConsent": auto_consent_value
})

cloud_account_patch_api_headers = {
 'accept': 'application/json',
 'content-type': 'application/json',
 'x-redlock-auth': YOUR_TOKEN
}

for cloud_type, account_ids in account_ids.items():
  for account_id in account_ids:
    response = requests.request(
      method="PATCH",
      url=cloud_account_patch_api_url.format(base_url=base_url,
cloudType=cloud_type, accountId=account_id),
      headers=cloud_account_patch_api_headers,
      data=cloud_account_patch_api_payload,
      verify=False
    )

    if response.status_code == 200:
      print(f"Successfully updated account ID - {account_id}")
    else:
      print(f"Failed to update account ID {account_id}, status_code:
{response.status_code}")
      print(f"Failed to update account ID {account_id}, response:
{response.text}")
      print(f"Failed to update account ID {account_id}, status:
{response.headers.get('x-redlock-status')}")
```

Once the disable script successfully executes, the **Prisma Cloud role** on CSP will now have only granular permissions and will **not** have read-all type permissions.

The cloud account status will be green until there are any missing permissions when new API ingestions are supported in the future.

Feedback and Support

We want your feedback! If you have any questions or clarifications while running the scripts or any of the steps listed in the document, please reach out to your Prisma Cloud Customer Success Representative or email at racharya@paloaltonetworks.com.