myob everyone's business

PayGlobal Data-on-demand

Guide to Integration

Prepared For

PayGlobal Customers

Produced By

PayGlobal Solutions & DevOps Team

20 November 2025

Version 2.0

Overview

This guide explains how customers can authenticate and programmatically access their exported PayGlobal data from their allocated S3 tenant folder using the AWS SDK and Amazon Cognito. It includes required login details, code examples, and security best practices.

Login Details Provided by MYOB

The PayGlobal Platform team will supply the following credentials and identifiers that you will use to authenticate and access your data. Store these securely and never hard-code them in source code or share them publicly.

Capability	Summary
user_pool_id	Provided by MYOB for Amazon Cognito user pool identification. Store securely and use for authentication workflows.
identity_pool_id	Provided by MYOB for Amazon Cognito identity pool mapping and session credential exchange.
app_client_id	Provided by MYOB for the Cognito app client configuration. Never hard-code; **keep in your secrets manager**.
app_client_secret	Provided by MYOB; **treat as a secret**. Never embed in code or share.
username	Your tenant-specific username for authentication. Also used as the default S3 prefix to scope your data.
password	Your tenant password for authentication. **Treat as a secret** and rotate regularly.
bucket_name	The target S3 bucket where your PayGlobal data is written. Your tenant data lives under your folder/prefix.

Then you will need to develop some code using the AWS SDK to retrieve the data from the tenant folder within the bucket.

DevOps have created a ‘login library’ file, written in Python, using Boto3 (AWS' SDK for Python), that handles the login and credential-grabbing process for customers. You may wish to continue to build on this solution for your integration, or use it as a reference when creating it (say, if you wanted to integrate in another language, instead of Python).

That library file is available here:
https://github.com/MYOB-Technology/pgol-data-cdk/blob/main/pgol_data_cdk/pipelines/post_deploy_tests/cognito_login.py

Prerequisites and Security Best Practices

- Python 3.13 minimum → Python Releases for Windows
- Python boto3 package → pip install boto3

When integrating with PayGlobal Data-Out, it is the customer's responsibility to securely store all secret information required for authentication and access. This includes, but is not limited to: app_client_id, app_client_secret, username, and password. Failure to properly protect these credentials can result in unauthorised access or compromise of sensitive data.

We strongly recommend the following practices for storing and managing secret credentials:

- **Use enterprise-grade secret management solutions** (e.g., AWS Secrets Manager, Azure Key Vault, or HashiCorp Vault) to store credentials instead of configuration files or environment variables where possible.
- Restrict access to secrets using **least privilege principles**. Only allow trusted processes and personnel to view or use credential information.
- **Never embed, hard-code, or share secrets** directly in code repositories, especially those accessible to wider teams or the public.

Instructions

Authenticate

You can log in by importing the cognito file and calling the perform_login() function.

For example:

On a command line, on the same folder as the downloaded **cognito_login.py** file

```
# # CODE BLOCK 1
from cognito_login import perform_login


# tenant_information = <some secure secret store retrieval here>
tenant_information = {}


# Extract required credentials
user_pool_id = tenant_information.get('user_pool_id')
identity_pool_id = tenant_information.get('identity_pool_id')
app_client_id = tenant_information.get('app_client_id')
app_client_secret = tenant_information.get('app_client_secret')
username = tenant_information.get('username')
password = tenant_information.get('password')
```

```
bucket_name = tenant_information.get('bucket_name')


authenticated_boto3_session = perform_login(user_pool_id, identity_pool_id, app_client_id,
app_client_secret, username, password)
```

Query the data

From here, now that you are authenticated, you can programmatically query and export the data, for example:

```
#CODE BLOCK 2
s3_client = authenticated_boto3_session.client('s3')


s3_client.list_objects_v2(Bucket=bucket_name, Prefix=f'{username}/',MaxKeys='2')
s3_client.get_object(Bucket=bucket_name, Key=f"{prefix}example-object.txt")
s3_client.download_file(Bucket=bucket_name, Key=f"{prefix}example-object.txt",
Filename='example-object.txt')
```

Further, you may want to explore the AWS SDK documentation; boto3's S3 method documentation is available here:
S3 - Boto3 1.40.69 documentation

Example

- Python with boto3 package installed
- Credentials received
- File **cognito_login.py** downloaded
- Command line open in the same location as the **cognito_login.py**
- Start a python session, just type python and press Enter
- Run code from CODE BLOCK 1
- After you execute the last command of the block (), you should get the message:

```
>>> authenticated_boto3_session = perform_login(user_pool_id, identity_pool_id, app_client_id, app_client_secret, username, password)
ID token obtained.
Temporary AWS credentials obtained, expiring 2025-11-18 11:29:27+11:00.
```

- Then, you need to start a s3_client session → *s3_client = authenticated_boto3_session.client('s3')*
- From now on, you can use the s3_client with the boto3 service as per documentation: S3 - Boto3 1.40.69 documentation
- This client enables you to do several actions that are restricted to READ access
- Also, all access requires the passing of the parameters Bucket=bucket_name and Prefix=f'{username}/', that is, the path where your credentials will allow access
- The list_objects_v2 will show a limited list of files; it needs to be paginated
- The default result are in a JSON with a list of files under "Contents"

```
{
  "ResponseMetadata":{
    "RequestId":"4RYQ5NEBJ52E1D07",

"HostId":"PQflRJXigPNPMKoNvleNZiFRy3SENnzW+34V7+qzRwB+daQgDWrWtWLr8x3+7kYsfqSL4EMcbfo=",
    "HTTPStatusCode":200,
    "HTTPHeaders":{
      "x-amz-id-2":"PQflRJXigPNPMKoNvleNZiFRy3SENnzW+34V7+qzRwB+daQgDWrWtWLr8x3+7kYsfqSL4EMcbfo=",
      "x-amz-request-id":"4RYQ5NEBJ52E1D07",
      "date":"Mon, 17 Nov 2025 23:38:56 GMT",
      "x-amz-bucket-region":"ap-southeast-2",
      "content-type":"application/xml",
      "transfer-encoding":"chunked",
      "server":"AmazonS3"
    },
    "RetryAttempts":0
  },
  "IsTruncated":true,
  "Contents":[
    {
      "Key":"pg1000nz/csv/deltas/2025/10/31/actualsconfirmation/part-00000-0d542614-0621-4ec5-9ce0-c35f1b1078d1-c000.csv",
      "LastModified":datetime.datetime(2025,10,31,9,8,21,"tzinfo=tzutc())",
      "ETag":"\"38fb813f35a67b20fa6338d8f8a674f2-1\"",
      "ChecksumAlgorithm":[
        "CRC64NVME"
      ],
      "Size":114,
      "StorageClass":"STANDARD"
    },
    {
      "Key":"pg1000nz/parquet/deltas/2025/10/31/actualsconfirmation/part-00000-c2bca85e-a193-45c5-8db7-669fb212a15f-c000.parquet",
      "LastModified":datetime.datetime(2025,10,31,4,51,56,"tzinfo=tzutc())",
      "ETag":"\"38fb813f35a67b20fa6338d8f8a674f2-1\"",
      "ChecksumAlgorithm":[
        "CRC64NVME"
      ],
```

```
        "Size":114,
        "StorageClass":"STANDARD"
      }
   ],
   "Name":"pgol-data-final-prod",
   "Prefix":"pg1000nz/",
   "MaxKeys":2,
   "EncodingType":"url",
   "KeyCount":2,

"NextContinuationToken":"10mIlU0AoJ83qx1r9M+IPJ5fYuopCPzYC4Eo4DEXYpNm5qd/gAos5/9cONXJOraNrVLKcLqTj5uUgMRGR6cOK1hOt2yllDVcWYfpkArgXMehTxq2KEqvHly480T55SqStmufKZKEcw72PiHhFrFWSR1wwYbRoXa7LpT+4RVbLy3l8gEb9slEN7rzEjB6MSaVz"
}
```


The code below can be used to list all files:

```
continuation_token = None
results = []
while True:
   # Build request parameters
   params = {
      'Bucket': bucket_name,
      'Prefix': f'{username}/'
   }
   # Add continuation token if present
   if continuation_token:
      params['ContinuationToken'] = continuation_token
   # Make the API call
   response = s3_client.list_objects_v2(**params)
   # Process contents if present
   if 'Contents' in response:
      for obj in response['Contents']:
         key = obj['Key']
         last_modified = obj['LastModified'].strftime('%Y-%m-%d %H:%M:%S')
         result_str = f"{key} - {last_modified}"
         results.append(result_str)
         print(result_str)
   # Check if there are more results
   if response.get('IsTruncated', False):
      continuation_token = response['NextContinuationToken']
   else:
      break
```

Below is the output of the script:

```
pg1000nz/csv/deltas/2025/11/03/employeehistory/part-00000-670385e0-faa4-47ba-870c-565c88d1d08c-c000.csv - 2025-11-03 10:08:17
```

```
pg1000nz/csv/deltas/2025/11/03/employeehistoryudf/part-00000-dda18cf2-276d-4b09-b3b3-87ccb0687b40-c000.csv - 2025-11-03 09:47:14

pg1000nz/csv/deltas/2025/11/03/employeekiwisaverstate/part-00000-c0602d8d-e1b2-4781-acd0-60b66be30005-c000.csv - 2025-11-03 09:29:58

pg1000nz/csv/deltas/2025/11/03/employeenote/part-00000-95353d2a-16d8-4b7a-b5ab-d462fc9ab438-c000.csv - 2025-11-03 09:07:13

pg1000nz/csv/deltas/2025/11/03/employeerate/part-00000-0676dc22-73f8-4512-8fde-940d93fac9b2-c000.csv - 2025-11-03 09:09:23

pg1000nz/csv/deltas/2025/11/03/employeetimebandoverride/part-00000-b0496123-3f5a-46b6-a5a5-4c773676107c-c000.csv - 2025-11-03 09:32:08

pg1000nz/csv/deltas/2025/11/03/employeetour/part-00000-32a853eb-de6e-4118-a25e-d172b0996243-c000.csv - 2025-11-03 09:06:58

pg1000nz/csv/deltas/2025/11/03/employeevisa/part-00000-5a36b8b2-0381-446b-aa35-a78b2c350587-c000.csv - 2025-11-03 10:04:32

pg1000nz/csv/deltas/2025/11/03/emplsuperfund/part-00000-5c3e8013-5849-48bc-a86e-3d2c4e1bfe18-c000.csv - 2025-11-03 09:09:13

pg1000nz/csv/deltas/2025/11/03/historicalactualallowance/part-00000-27746dda-5eb3-4925-beea-5b85171129b0-c000.csv - 2025-11-03 10:01:06

pg1000nz/csv/deltas/2025/11/03/historicalactualtimeband/part-00000-4a85aa20-b47b-4e52-864f-b2772f8ea29d-c000.csv - 2025-11-03 09:11:45

pg1000nz/csv/deltas/2025/11/03/historicalallowance/part-00000-665223a0-7135-4a2a-a0fa-1c23e69a7b54-c000.csv - 2025-11-03 10:12:39

pg1000nz/csv/deltas/2025/11/03/historicalauditlog/part-00000-44e75fc8-3fdf-4ca2-9382-0a03cd6d8dfd-c000.csv - 2025-11-03 09:38:48

pg1000nz/csv/deltas/2025/11/03/historicaldeduction/part-00000-d5d7425b-1e5b-400c-9753-6b2cbcfd952a-c000.csv - 2025-11-03 09:40:12

pg1000nz/csv/deltas/2025/11/03/historicalproportionalcost/part-00000-6907eaf4-9514-4f4f-95d0-f1611d9fecf7-c000.csv - 2025-11-03 09:3
```

On the code block example, you scan filter by types and or dates. On the line 7='Prefix': f'{username}/', you can include variations for the request:

- All delta csv files for the day: 'Prefix': f'{username}/csv/deltas/2025/11/03'
- All delta parquet files for the day: 'Prefix': f'{username}/parquet/deltas/2025/11/03'
- All full snapshot csv files for a table: 'Prefix': f'{username}/csv/full/employee/'

Error Handling and Support

- **Authentication failures**: Verify all login details match those provided by MYOB and confirm they're retrieved securely at runtime. Consider token-based auth to reduce repeated credential handling.
- **Access denied**: Ensure IAM permissions for the Cognito-authenticated role permit S3 access to your tenant prefix. Confirm you are using the correct bucket_name and prefix.
- **Data not found**: Confirm you are listing the correct tenant prefix (often your username) and check your data export schedule or naming conventions.
- **Support**: Contact your MYOB representative with your tenant identifier, bucket_name, and steps taken so far. Provide timestamps, error messages, and any request IDs to speed up investigation.

Frequently Asked Questions (FAQs)

1. Can we implement the integration in languages other than Python?

Yes. The Python login helper is provided for convenience; you can use it as a reference to build equivalent logic in other languages.

2. Is there a way to reduce risk from storing raw credentials?

Use token-based authentication: retain an ID token and refresh token after initial authentication, and use the refresh token to obtain new access tokens without re-sending raw credentials. Secure and rotate tokens as needed.

3. What happens if a system operation fails?

The system automatically retries failed processes while notifying DevOps of any issues, ensuring reliability.

4. Where in the bucket is our data?

Your data is in your tenant folder/prefix, commonly using your username as the S3 prefix under the provided bucket_name.

5. Do we need Boto3 specifically?

The guide demonstrates Boto3 in Python, but any AWS SDK that supports Cognito auth and S3 access is acceptable. Use the Python helper as a reference or adopt the same flow in your chosen language.