Python SDK to upload mail to Vaultastic Active Store

Overview

This document contains the instructions to use the Python SDK to upload data to the Vaultastic Active Store using the Vaultastic Mail Upload API.

Prerequisites

1. You should have your Vaultastic domain provisioned.
2. You should have created the vaults in the Active Store to which you want to upload the data
3. You have got the API key from Mithi.
4. You should have Python version 3.6 or above.
5. The following are the dependencies:
    - argparse
    - io
    - json
    - logging
    - sys
    - requests
    - socket
    - time
    - uuid
    - email
6. The following environment variables should be set
    a. PRIMARY_DOMAIN: The primary domain name for filtering vaults from the mail.
    b. API_ENDPOINT_URL: ActiveStoreMailUploadAPI.vaultastic.com
    c. API_KEY: The API Key shared for uploading mail to v4.

Uploading mail

To upload mail to a vault in the Active Store of your Vaultastic domain use the script **VaultasticActiveStoreMailUpload.py**

The command line arguments are as follows:

1. -i where is the path of the EML file.
2. -m where are the mail contents
3. -r where are the comma separated list of ids on the primary domain
4. -s where the sender is an id on the primary domain

Important Note:

1. Use either -i or -m option but not both. If both are used the -m value will be ignored.
2. Use -r and -s only if the sender or recipient is not part of the mail headers From, To, CC or BCC

VaultasticActiveStoreMailUpload-API-SDK Exit Codes and Messages

Exit Code	Description	Resolution
0	Mail Successfully uploaded	N.A.
1	Unknown Exception	This is the exit code returned when there is no other specific exit code defined for the error that occurred. More Information can be found in the message/stack trace printed by python.
2	Misuse of shell builtins (UNIX reserved error)	Missing keyword or command, or permission problem (and diff return code on a failed binary file comparison)
3	Invalid number of arguments	invalid number of arguments passed. Provide at least -i or -m.
4	Invalid Recipient	Invalid Recipient Email ID. Make sure the id passed on -r argument is a valid email id (or email id's are comma separated).
5	Failed to Upload mail to S3	Mail Failed to upload to s3 bucket. Check Error logs for the failed message.
6	Recipients not from PRIMARY_DOMAIN	At least one recipient should be from the primary domain. 1. Check if PRIMARY_DOMAIN is correct OR 2. Check any of the recipients in the mail (from, to, cc or bcc headers) is from the PRIMARY_DOMAIN. OR 3. Provide a recipient id using the -r argument.

Limits

1. No multi-domain support. That is recipient filtering based on multiple PRIMARY_DOMAIN values is not supported.
2. Rate Limits: 10,000 Requests per second with 5000 Burst requests.
3. Max Mail Size: 50MB

Resources

- VaultasticActiveStoreMailUpload-SDK-1.0.2.zip (http://vaultastic.mithi.com/res/VaultasticActiveStoreMailUpload-SDK-1.0.2.zip)

Example Implementations

Method 1: Implementation of VaultasticActiveStoreMailUpload-SDK on SNS triggered Lambda using subprocess

Pre-Requisites

1. Memory Required: 128MB (for up to 10MB Mails)

2. Lambda Timeout: 3 mins (180 seconds)

3. Runtime: Python 3.6

4. Handler: lambda_function.lambda_handler

5. Environment Variables:

   a) API_ENDPOINT_URL: https://vaultasticactivestoremailuploadapi.vaultastic.com

   b) API_KEY:

   c) PRIMARY_DOMAIN:

6. Make sure Lambda has Internet access. (for VPC Lambda see this: https://aws.amazon.com/premiumsupport/knowledge-center/internet-access-lambda-function/)

Sample code to run VaultasticActiveStoreMailUpload-SDK on SNS triggered Lambda :

1. Using the sample code given create a Python script (lambda_function.py)

```
import subprocess
```

```
def lambda_handler(event, context):
    #assuming lambda is triggerd by SNS
    #extract message from SNS event
    message = event['Records'][0]['Message']

    filepath = '/tmp/uploadmailToV4.eml'

    file1 = open(filepath,'w')
    #write message to file
    file1.write(message)
    file1.close()

    #call VaultasticActiveStoreMailUpload.py as subprocess
    exit_code  = subprocess.call("python VaultasticActiveStoreMailUpload.py -i "+filepath, shell=True)
    print(exit_code)
```

2. Add this lambda_function.py to the VaultasticActiveStoreMailUpload-SDK.zip.

3. Upload zip to Lambda Function with the above parameters.

Method 2: Implementation of VaultasticActiveStoreMailUpload-SDK without using the subprocess module on S3 triggered Lambda:

Pre-Requisites

1. Memory Required: 128MB (for up to 5MB Mails) - more memeory required as the mail content resides in the same process

2. Lambda Timeout: 3 mins (180 seconds)

3. Runtime: Python 3.6

4. Handler: lambda_function.lambda_handler

5. Environment Variables:

a) API_ENDPOINT_URL: https://vaultasticactivestoremailuploadapi.vaultastic.com

b) API_KEY:

c) PRIMARY_DOMAIN:

6. Make sure Lambda has Internet access. (for VPC Lambda see this: https://aws.amazon.com/premiumsupport/knowledge-center/internet-access-lambda-function/)

Sample code to run VaultasticActiveStoreMailUpload-SDK without a subprocess:

1. Using the sample code given create a Python script (lambda_function.py)

```
import os
import boto3
import io
from VaultasticActiveStoreMailUpload import uploadMailToVaultasticActiveStore

def lambda_handler(event, context):
    #assuming lambda is triggerd by S3
    #extract message from S3 event
    bucketName = event['Records'][0]['s3']['bucket']['name']
    objectPath = event['Records'][0]['s3']['object']['key']

    s3Client = boto3.client('s3')
    s3Object = s3Client.get_object(Bucket=bucketName,Key=objectPath)

    message = s3Object['Body'].read().decode()

    uploadMailToVaultasticActiveStore(message)
```

2. Add this lambda_function.py to the VaultasticActiveStoreMailUpload-SDK.zip.

3. Upload zip to Lambda Function with the above parameters.

4. Send a mail using SNS and it should upload to v4-s3 with exit code 0.

Choosing the right method

The method to choose depends on a number of parameters as highlighted below

Parameter	Method 1: Subprocess method	Method 2: Function method
Target vault	Has to be used when the target vault is not part of the mail header (from, to, cc or bcc field)	Can only be used when the target vault is part of the mail header

Mail body	Has to be used when the mail content is fetched from another process	Can only be used when mail content is available locally
Error handling	Has to be used for comprehensive error handling	Can be used when only success or failure has to be checked
Performance requirements	Can take upto 6-15 seconds depending on the mail size	Can take 2-15 seconds depending on mail size
Memory requirements	Lambda with 128 MB can handle mail upto 10 MB	Lambda with 128 MB can handle mail upto 5 MB (as mal content are part of the process)
Programming expertise	Moderate	High