ASSOCIATION OF PUBLIC HEALTH LABORATORIES

APHL Informatics Messaging Services (AIMS)

S3 Transport *NIX CLI Configuration Guide - Version 2 -

Contents

APHL Informatics Messaging Services (AIMS) Platform

The APHL Informatics Messaging Services (AIMS) platform is a secure, cloud based environment that accelerates the implementation of public health messaging solutions by providing shared services to aid in the transport, validation, translation and routing of electronic data.

AIMS is located on Amazon Web Services (AWS) East, West and GovCloud regions. Multi-availability zones are also in use.

Transport Compatibility and Interoperability include:

- PHINMS
- Direct
- SFTP
- Web Services
- S3

This document details the steps to install and configure the AIMS S3 Client.

Amazon Web Services (AWS) S3

Amazon Simple Storage Service (Amazon S3), provides developers and IT teams with secure, durable, highly-scalable object storage. Amazon S3 is easy to use, with a simple web services interface to store and retrieve any amount of data from anywhere on the web. With Amazon S3, you pay only for the storage you actually use. There is no minimum fee and no setup cost.

Amazon S3 can be used alone or together with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), and Amazon Glacier, as well as third party storage repositories and gateways. Amazon S3 provides cost-effective object storage for a wide variety of use cases including cloud applications, content distribution, backup and archiving, disaster recovery, and big data analytics. Key capabilities and benefits include:

- Amazon S3 provides durable infrastructure to store important data and is designed for durability of 99.999999999% of objects. Your data is redundantly stored across multiple facilities and multiple devices in each facility.
- Amazon S3 is designed for 99.99% availability of objects over a given year. Amazon S3 is also backed by the Amazon S3 Service Level Agreement, ensuring that you can rely on it when you need it. And you can choose an AWS region to optimize for latency, minimize costs, or address regulatory requirements.
- Amazon S3 supports data transfer over SSL and automatic encryption of your data once it is uploaded. You can also configure bucket policies to manage object permissions and control access to your data using AWS Identity and Access Management (IAM).
- With Amazon S3, you can store as much data as you want and access it when you need it. You can stop guessing your future storage needs and scale up and down as required, dramatically increasing business agility.
- Amazon S3 can send event notifications when objects are uploaded to Amazon S3. Amazon S3 event notifications can be delivered using Amazon SQS or Amazon SNS, or sent directly to AWS Lambda, enabling you to trigger workflows, alerts, or other processing. For example, you could use Amazon S3 event notifications to trigger transcoding of media files when they are uploaded, processing of data files when they become available, or synchronization of Amazon S3 objects with other data stores.
- Amazon S3 supports multi-part uploads to help maximize network throughput and resiliency, and lets you choose the AWS region to store your data close to the end user and minimize network latency. And Amazon S3 is integrated with Amazon CloudFront, a content delivery web service that distributes content to end users with low latency, high data transfer speeds, and

no minimum usage commitments.

- Amazon S3 is integrated with other AWS services to simplify uploading and downloading data from Amazon S3 and make it easier to build solutions that use a range of AWS services. Amazon S3 integrations include Amazon CloudFront, Amazon Kinesis, Amazon RDS, Amazon Glacier, Amazon EBS, Amazon DynamoDB, Amazon Redshift, Amazon Route 53, Amazon EMR, and AWS Lambda.
- Amazon S3 is easy to use with a web-based management console and mobile app and full REST APIs and SDKs for easy integration with third party technologies.

AIMS uses several S3 clients. One example is the CrossFTP solution to facilitate secure transport from your organization to AIMS and to other trading partners on AIMS.

AIMS S3 *NIX Client Installation Guide

This guide was prepared for Amazon Linux AMI 2015.03. You may need to make several adjustments based on the flavor of Linux/UNIX you choose and your specific IT policies and procedures.

This guide will help you:

1. Download and install the AWS CLI Client for Python.
2. Validate that the basic client works.
3. Configure the AWS CLI client for the AIMS Platform S3 data exchange.
4. Test to ensure that the installation and configuration works and allows access to the appropriate bucket on S3.
5. Download and install the CrossFTP client (in order to access the decryption routines on the commandline).
6. Create the aimsplatformfetch.sh shell script that will manage the retrieval and decryption of data from the AIMS Platform.
7. Test the aimsplatformfetch.sh script on the commandline.
8. Schedule the aimsplatformfetch.sh script for regular execution (every minute or 5 minutes).

Download and Install AWS CLI

The AWS CLI may already be installed on your environment. If it is not you can retrieve it here:
http://docs.aws.amazon.com/cli/latest/userguide/installing.html#install-bundle-other-os

Use the following command to retrieve the AWS CLI from the default location where Amazon hosts it:

```
[ec2-user@ip-private ~]$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-
bundle.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 5072k  100 5072k    0     0  17.1M      0 --:--:-- --:--:-- --:--:-- 17.2M
[ec2-user@ip-private ~]$
```

Unzip the AWS CLI to prepare for installation:

```
[ec2-user@ip-private ~]$ unzip awscli-bundle.zip
Archive:  awscli-bundle.zip
  inflating: awscli-bundle/install
  inflating: awscli-bundle/packages/argparse-1.2.1.tar.gz
  inflating: awscli-bundle/packages/awscli-1.7.22.tar.gz
  inflating: awscli-bundle/packages/bcdoc-0.13.0.tar.gz
  inflating: awscli-bundle/packages/botocore-0.103.0.tar.gz
  inflating: awscli-bundle/packages/colorama-0.3.3.tar.gz
  inflating: awscli-bundle/packages/docutils-0.12.tar.gz
  inflating: awscli-bundle/packages/jmespath-0.6.1.tar.gz
  inflating: awscli-bundle/packages/ordereddict-1.1.tar.gz
  inflating: awscli-bundle/packages/pyasn1-0.1.7.tar.gz
  inflating: awscli-bundle/packages/python-dateutil-2.4.2.tar.gz
  inflating: awscli-bundle/packages/rsa-3.1.4.tar.gz
  inflating: awscli-bundle/packages/simplejson-3.3.0.tar.gz
  inflating: awscli-bundle/packages/six-1.9.0.tar.gz
  inflating: awscli-bundle/packages/virtualenv-1.10.1.tar.gz
[ec2-user@ip-private ~]$
```

Install the AWS CLI for the current user using the command below:

```
[ec2-user@ip-private ~]$ ./awscli-bundle/install -b ~/bin/aws
Running cmd: /usr/bin/python virtualenv.py --python /usr/bin/python /home/ec2-
user/.local/lib/aws
Running cmd: /home/ec2-user/.local/lib/aws/bin/pip install --no-index --find-links
file:///home/ec2-user/awscli-bundle/packages awscli-1.7.22.tar.gz
You can now run: /home/ec2-user/bin/aws --version
[ec2-user@ip-private ~]$
```

Validate Basic AWS CLI Functionality

This validates that the AWS CLI is in fact able to run. Your output should be similar, but Python version and CLI versions may differ:

```
[ec2-user@ip-private ~]$ /home/ec2-user/bin/aws --version
aws-cli/1.7.22 Python/2.7.9 Linux/3.14.35-28.38.amzn1.x86_64
[ec2-user@ip-private ~]$
```

Configure the AWS CLI

In order for the commands to work as expected it may be necessary to have the ~/bin folder in your path. Please add it as is appropriate for your distribution by editing the .profile .bash_rc file in your home directory.

```
Next run the configuration script for the AWS CLI in order to store a profile that can be used
```

```
by the aws command.  The authentication information is provided separately.
[ec2-user@ip-private bin]$ /home/ec2-user/bin/aws configure --profile aimsplatform
AWS Access Key ID [None]: <access key id>
AWS Secret Access Key [None]: <secret access key>
Default region name [None]: fips-us-gov-west-1
Default output format [None]: Text
[ec2-user@ip-private bin]$
```

Test S3 Access

Test your systems ability to interact with the S3 bucket associated with you by executing the following command (the account id will provided separately):
aws s3 ls s3://aimsplatformgov/data/**<account id>**/ --profile aimsplatform

Download and Install CrossFTP

Download the command line version of the CrossFTP client (provided the ability to decrypt files that were encrypted with CrossFTP Windows Client):

wget http://www.crossftp.com/crossftp-all-bin-1.96.6_decrypt.zip

Unzip the CrossFTP command client client using the command below. This command will ensure that all the extracted files are placed in the ~/bin folder. This ensures the aimsplatformfetch.sh command works as intended and that no changes are needed to the decrypt.sh command provided by CrossFTP.

unzip -j crossftp-all-bin-1.96.6_decrypt.zip crossftp-all-bin/* -d ~/bin/

Create the aimsplatformfetch.sh Shell Script

Create the aimsplatformfetch.sh script in the ~/bin folder. A digital copy will also provided with the credentials. Please replace the account id and decryption password with the values that were provided by the AIMS Platform team.

```
#!/bin/bash

PASSWORD="<decryption password>"
ACCOUNTID="<account id>"
AIMSHOME="${HOME}/aimsplatform"
AIMSLOCK="${AIMSHOME}/.lock"
CROSSFTPINSTALL="${HOME}/bin"

echo Checking for files  - `date`

# Check for lock file
if [[ -f "${AIMSLOCK}" ]]; then
        # There was a lock file ... the process is running already (probably with a large file fetch)
        echo " * A lock file exists - we will exit and wait for the next scheduled time *"
        exit 0;
fi

# Create a lock file
touch ${AIMSLOCK}

# Please note - this utility does not handle files with spaces in the names correctly at present.  It is
being addressed.
# The decryption utility has problems handling spaces

aws s3 mv s3://aimsplatformgov/data/${ACCOUNTID}/ReceivedFrom/ ${AIMSHOME} --recursive --profile
aimsplatform
find  ${AIMSHOME} -name "*.aes" >  ${AIMSHOME}/.filelist

for i in `cat ${AIMSHOME}/.filelist`
do
        echo "Processing - $i"
        echo " - Prevalidation"
        fbnoaes=${i%.aes}
        if [[ -f "${fbnoaes}" ]]; then
```

```
                # Prevent the file from being decrypted if it is already there
                echo " - ERROR: Decrypted file is already present in the folder"
                continue;
        fi
        echo " - Decrypting"
        java -cp ${CROSSFTPINSTALL}/crossftp.jar:${CROSSFTPINSTALL}/crossftp-
resources.jar:${CROSSFTPINSTALL}/commons-logging.jar:${CROSSFTPINSTALL}/jnlp.jar:${CROSSFTPINSTALL}/looks-
2.2.1.jar:${CROSSFTPINSTALL}/bcprov-jdk15on-147.jar crossftp.common.DecryptionUtil ${PASSWORD} "${i}"
        echo " - Decrypted $fbnoaes"
        echo " - Validating that $fbnoaes exists"
        if [[ ! -f "${fbnoaes}" ]]; then
                # Prevent the file from being deleted if it is not decrypted
                echo " - ERROR: Decrypted file was not present"
        else
                # Delete only files that did decrypt
                echo " - Remove $i"
                rm -f "${i}"
        fi
        echo " - Complete"
done

# Destroy the lock file
rm -f ${AIMSLOCK}
```

Test aimsplatformfetch.sh

Ensure that the aimsplatformfetch.sh script is set to be executable by your user and test it by running it (the sample below shows the script in the ~/bin folder, our current folder is ~/bin). If files were already available for you to pick up the output will be more elaborate as shown in the next step:

```
[ec2-user@ip-private bin]$ ./aimsplatformfetch.sh
[ec2-user@ip-private bin]$
```

Ask the AIMS Platform Admins to place a file on the server, test the script again. The sample could look as is shown below, depended on who made files available and for what program/function:

[ec2-user@ip-private bin]$ ./aimsplatformfetch.sh
move: s3://aimsplatformgov/data/**<account id>**/ReceivedFrom/APHL-WI_WIDOH/Staging/AMD/text.txt.aes to ~/aimsplatform/APHL-WI_WIDOH/Staging/AMD/text.txt.aes
Processing - ~/aimsplatform/APHL-WI_WIDOH/Staging/AMD/text.txt.aes
 - Prevalidation

```
 - Decrypting
 - Decrypted ~/aimsplatform/APHL-WI_WIDOH/Staging/AMD/text.txt
 - Validating that ~/aimsplatform/APHL-WI_WIDOH/Staging/AMD/text.txt exists
 - Remove ~/aimsplatform/APHL-WI_WIDOH/Staging/AMD/text.txt.aes
 - Complete
[ec2-user@ip-private bin]$
```

Schedule aimsplatformfetch.sh for frequent execution

In order for you to be able to fetch files automatically, in the background, you can schedule the script to run every minute or any interval of your choosing. The script it multi-execution safe (it will not fail or behave in unexpected ways if it is already running).

```
[ec2-user@ip-private bin]$ crontab -l
* * * * * /home/ec2-user/bin/aimsplatformfetch.sh >> /home/ec2-user/aimsplatform/.scheduledrun.log
[ec2-user@ip-private bin]$
```

Support

Eduardo Gonzalez Loumiet
eddie@uberops.com
850-766-5338

APHLHUB@UberOps.com | +1 (888) 404-5409