AWSParams Documentation

Nate Peterson

Nov 01, 2019

Contents:

CHAPTER 1

Note

Version 1 of this library is drastically different than previous versions. The CLI Application hasn't changed but the library it uses has. Please pay extra attention to the examples below or look at the underlying class for more information.

CHAPTER 2

Why this script?

The current (Jul 2017) AWS Console for the Systems Manager Parameter Store is good for adding and editing the values of parameters, but misses key productivity functions like copying (especially en mass), renaming, etc. The current `aws ssm` CLI is very similar in functionality to the AWS Console.

This script is to automate a lot of the manual work currently needed with the existing AWS-provided UIs.

CHAPTER 3

Installation

- AWSParams requires Python 3.6+
- Depending on your Python3.6 install either `pip install awsparams` or `pip3 install awsparams`

CHAPTER 4

Usage

4.1 Library:

```
from awsparams import AWSParams

# Using default Profile
aws_params = AWSParams()

# Using a Custome Profile
aws_params = AWSParams('MyProfile')

#get a single parameter
param = get_parameter('test1')
# ParamResult(Name='test1', Value='test123', Type='SecureString')

#ParamResult is a named tuple with properties Name, Value, Type
param.Name # 'test1'
param.Value # 'test123'
param.Type # 'SecureString'

# get multiple parameters with a prefix
params = get_all_parameters(prefix="testing.testing.")
# [ParamResult(Name='testing', Value='1234', Type='String'),
#  ParamResult(Name='testing2', Value='1234', Type='String')]

# get multiple parameters by path
params = get_all_parameters(prefix="/testing/testing/", by_path=True)
# [ParamResult(Name='testing', Value='1234', Type='String'),
#  ParamResult(Name='testing2', Value='1234', Type='String')]

# get multiple parameters by path
params = get_all_parameters(prefix="/testing/testing/", by_path=True, trim_name=False)
# [ParamResult(Name='/testing/testing/testing', Value='1234', Type='String'),
#  ParamResult(Name='/testing/testing/testing2', Value='1234', Type='String')]
```

(continues on next page)

(continued from previous page)

```
# get just a parameter value
value = get_parameter_value('test1')
# test123
```

For more detailed examples of usage as a library see the cli implementation here.

For full library reference see: *here*.

4.2 CLI application:

Usage can be referenced by running `awsparams --help` or `awsparams subcommand --help` commands:

```
Usage: awsparams [OPTIONS] COMMAND [ARGS]...

Options:
--version  Show the version and exit.
--help     Show this message and exit.

Commands:
cp   Copy a parameter, optionally across accounts
ls   List Paramters, optional matching a specific...
mv   Move or rename a parameter
new  Create a new parameter
rm   Remove/Delete a parameter
set  Edit an existing parameter
```

More examples *here*

4.2.1 CLI application

Usage can be referenced by running `awsparams --help` or `awsparams subcommand --help` commands:

```
Usage: awsparams [OPTIONS] COMMAND [ARGS]...

Options:
--version  Show the version and exit.
--help     Show this message and exit.

Commands:
cp   Copy a parameter, optionally across accounts
ls   List Paramters, optional matching a specific...
mv   Move or rename a parameter
new  Create a new parameter
rm   Remove/Delete a parameter
set  Edit an existing parameter
```

Command Examples

ls usage

ls names only: `awsparams ls`

ls with values no decryption: `awsparams ls --values` or `awsparams ls -v`

ls with values and decryption: `awsparams ls --with-decryption`

ls by prefix: `awsparams ls appname.prd`

new usage

new interactively: `awsparams new`

new semi-interactively: `awsparams new --name appname.prd.username`

new non-interactive: `awsparams new --name appname.prd.usrname --value parameter_value --description parameter_descripton`

cp usage

copy a parameter: `awsparams cp appname.prd.username newappname.prd.username`

copy set of parameters with prefix appname.dev. to appname.prd.: `awsparams cp appname.dev. appname.prd. --prefix`

copy set of parameters starting with prefix repometa-generator.prd overwrite existing parameters accross different accounts: `awsparams cp repometa-generator.prd --src_profile=dev --dst_profile=trn --prefix=True`

copy single parameters accross different accounts: `awsparams cp appname.dev.username appname.trb.us`

4.2.2 AWSParams

class awsparams.**AWSParams**(*profile: str = ''*)

AWSParams handles all Parameter Store operations

Parameters **profile** (*optional*) – AWS Profile to use for the session

ssm

Boto3 SSM Client object

Type boto3.client

profile

AWS Profile to use for the session

Type str, optional

build_param_result(*param: dict, *, prefix: str = '', values: bool = True*) → awsparams.ParamResult

Build a parameter result

Parameters

- **param** (*dict*) – Parameter to build ParamResult for
- **prefix** (*str, optional*) – If passed prefix will be removed from parameter name
- **values** (*bool, optional*) – Flag to toggle values defaults True

Returns Parameter result in a ParamResult NamedTuple.

Return type *ParamResult*

get_all_parameters(*, *prefix: str = ''*, *values: bool = True*, *decryption: bool = True*, *trim_name: bool = True*) → List[awsparams.ParamResult]

Get all parameters Optionally by prefix or path

If prefix starts with a / then A Parameter path is assumed and the calls to aws will use path api's which are more performant than traversing all parameters

Parameters

- **prefix** (*str, optional*) – Prefix to filter parameters on
- **values** (*bool, optional*) – Flag toggle values defaults True
- **decryption** (*bool, optional*) – Flag to toggle decryption defaults True
- **trim_name** (*bool, optional*) – Flag to toggle name trimming on results defaults True

Returns List of Parameter Results

Return type List[*ParamResult*]

get_parameter(*name: str*, *, *values: bool = True*, *decryption: bool = True*) → Optional[awsparams.ParamResult]

Get a specific Parameter

Parameters

- **name** (*str*) – Name of parameter to get
- **values** (*bool, optional*) – Flag to toggle values defaults True
- **decryption** (*bool, optional*) – Flag to choose decryption. Defaults True

Returns The Parameter for success or else None

Return type *ParamResult*, None

get_parameter_value(*name: str*, *, *decryption: bool = True*) → str

Get a specified Parameter's Value

Parameters

- **name** (*str*) – Name of parameter to get
- **decryption** (*bool, optional*) – Flag to choose decryption. Defaults True

Returns Value of the Parameter as a string.

Return type str

new_param(*name: str*, *value: str*, *, *param_type: str = 'String'*, *key: str = ''*, *description: str = ''*, *overwrite: bool = False*)

Create a new parameter

Parameters

- **name** (*str*) – Name of new parameter.
- **value** (*str*) – Value of the new parameter
- **param_type** (*str, optional*) – Type of New parameter default "String"
- **key** (*str, optional*) – KMS Key to encrypt default ""
- **description** (*str, optional*) – Description of the new parameter default ""
- **overwrite** (*bool, optional*) – Flag to toggle overwriting existing parameter default False

put_parameter(*parameter: dict*, *, *overwrite: bool = False*, *profile: str = ''*)
Put a Parameter

Parameters

- **parameter** (*dict*) – Parameter to create
- **overwrite** (*bool, optional*) – Flag to overwrite existing parameters
- **profile** (*str, optional*) – Optional specifiy a alternate profile to use

remove_parameter(*param: str*)
Remove a Parameter

Parameters **param** (*str*) – Name of the parameter to remove

set_param(*param: str*, *value: str*) → bool
Edit an existing parameter

Parameters

- **param** (*str*) – Name of parameter to set.
- **value** (*str*) – Value to set.

Returns True for modified False for unmodified

Return type bool

4.2.3 ParamResult

class awsparams.**ParamResult**
ParamResult is a NamedTuple that represents a Parameter result.

Name
Name of the Parameter

Type str

Value
Value of the Parameter

Type str

Type
Type of the Parameter

Type str

CHAPTER 5

Indices and tables

- genindex
- modindex

Python Module Index

a

awsparams, 11

Index