User Guide: fn_aws_iam_v1.0.0

Table of Contents

Key Features

Amazon Web Services Identity and Access Management (AWS IAM) allows management of access to AWS services and resources securely. Using IAM, AWS users and groups can be created and managed, permissions can used to allow and deny access to AWS resources. The AWS IAM integration with the Resilient platform allows for querying and updating of users or access keys for an AWS account.

The following type of queries can be executed:

- Get a list of users and associated items (login profile, access keys, groups, policies).
- Get a list of access keys.

The integration can also be used to make the following changes to a SEP environment:

- Delete a user and delete or remove items associated with the user.
- Attach a user policy.
- Detach all policies for a user.
- Add user to a group.
- Remove a user from all groups.
- Change a user profile password.
- Delete an access key.
- Delete all access keys for a user.
- Delete the login profile for a user.

Function - AWS IAM: List Users

Get IAM user or users in the AWS account. Users can be filtered by user name , group and policy. If the user name is specified get information only for this user. Parameter aws_iam_user_name is an IAM user name. Parameters aws_iam_user_filter, aws_aim_group_filter and aws_aim_policy_filter param (all optional) are filters used to refine user data returned. Parameter aws_iam_query_type (optional) is used to determine type of query to perform users.

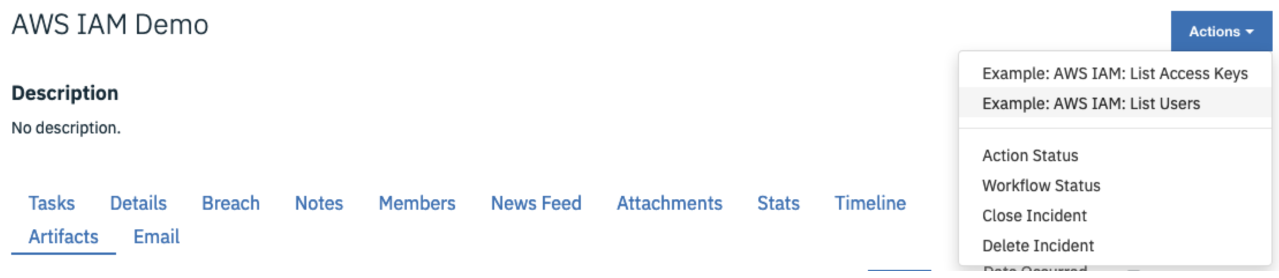
- Example workflows that use this Resilient Function include `Example: AWS IAM: List Users`, `Example: AWS IAM: List Access Keys`, `Example: AWS IAM: Refresh User`, `Example: AWS IAM: Delete Access Key For Artifact`, `Example: AWS IAM: Delete Login Profile`, `Example: AWS IAM: Delete User`, `Example: AWS IAM: Delete User For Artifact`, `Example: AWS IAM: Get Access Key For Artifact` and `Example: AWS IAM: Get User For Artifact`.

The workflow, `Example: AWS IAM: List Users`, sets the following input fields for the Function:

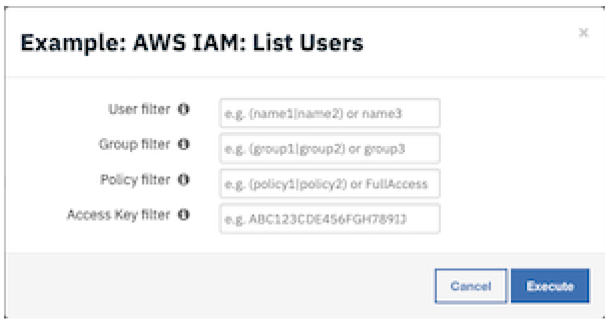
- aws_iam_user_filter (optional) is mapped to an activity field input. Note: Input should be a valid regular expression.
- aws_iam_group_filter (optional) is mapped to an activity field input. Note: Input should be a valid regular expression.
- aws_iam_policy_filter (optional) is mapped to an activity field input. Note: Input should be a valid regular expression.
- aws_iam_access_key_filter (optional) is mapped to an activity field input. Note: Input should be a valid regular expression.
- aws_iam_query_type is set to value `users`.

The workflow is initiated by the incident rule, `Example: AWS IAM: List Users`.

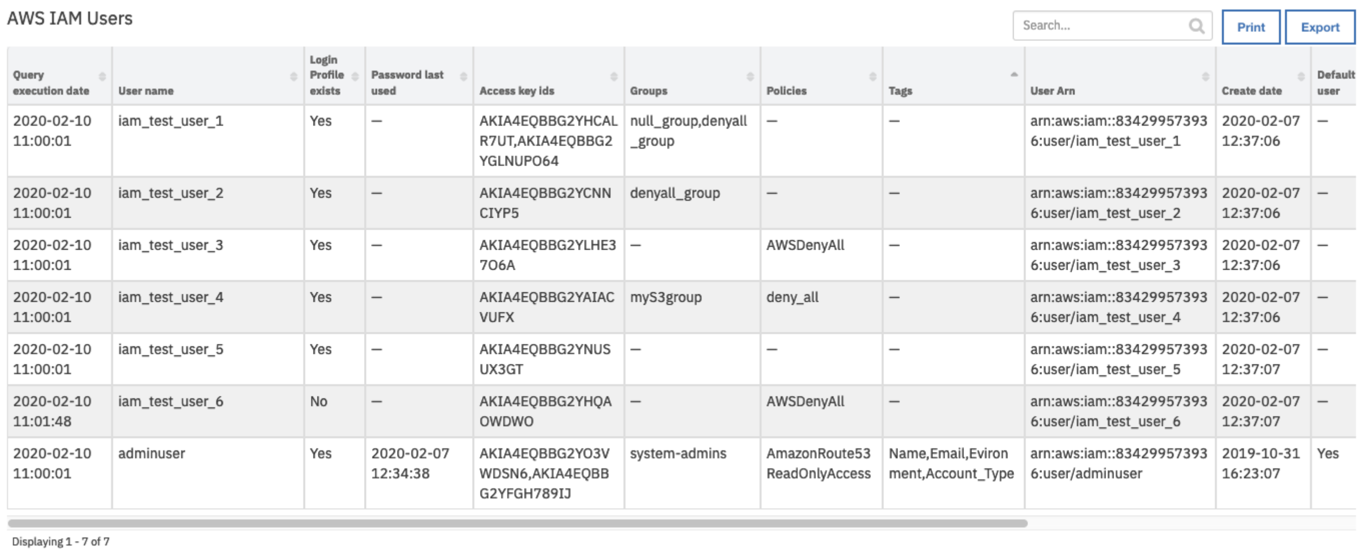
1. Open an incident and select `Example: AWS IAM: List Users` from Actions.



2. The user is presented with a list of input fields which can be used to filter users using regular expressions. Set any desired filters and click Execute..

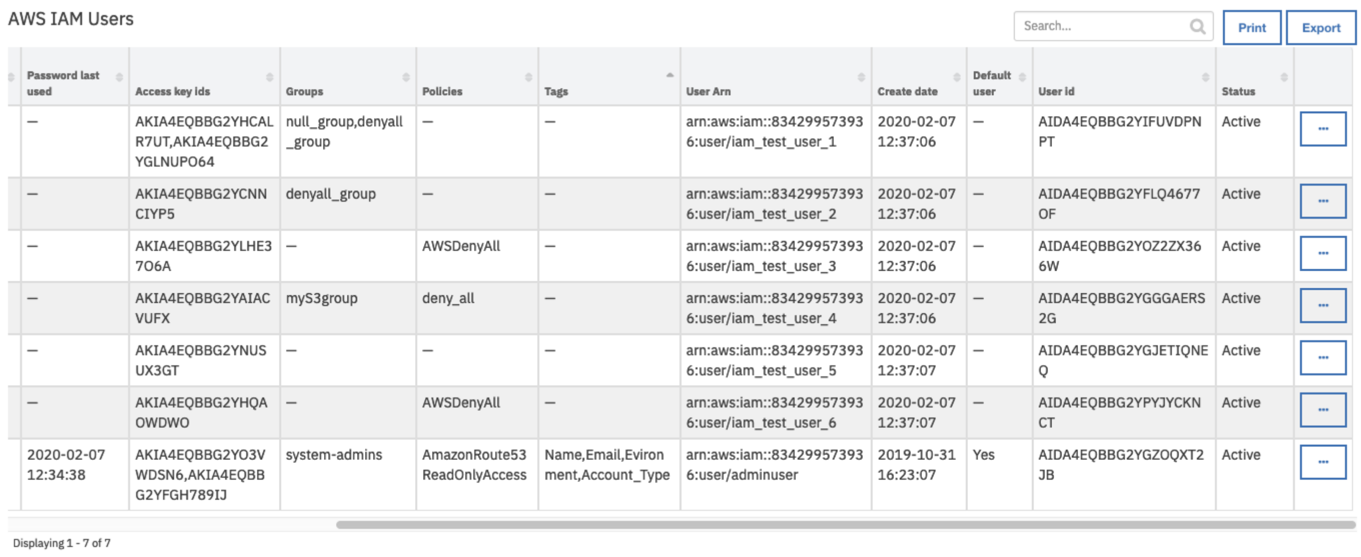

Example: AWS IAM: List Users
User filter
e.g. (name1|name2) or name3
Group filter
e.g. (group1|group2) or group3
Policy filter
e.g. (policy1|policy2) or FullAccess
Access Key filter
e.g. ABC123CDE456FGH789IJ
Cancel
Execute

This invokes the `Example: AWS IAM: List Users` workflow, which calls the `AWS IAM: List Users` function. The data table `AWS IAM Users` will be updated in the Resilient platform with the users properties for the selected AWS account.


AWS IAM Users

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user
2020-02-10 11:00:01	iam_test_user_1	Yes	—	AKIA4EQBBG2YHCALR7UT,AKIA4EQBBG2YGLNUPO64	null_group,denyall_group	—	—	arn:aws:iam::834299573936:user/iam_test_user_1	2020-02-07 12:37:06	—
2020-02-10 11:00:01	iam_test_user_2	Yes	—	AKIA4EQBBG2YCNNCIYP5	denyall_group	—	—	arn:aws:iam::834299573936:user/iam_test_user_2	2020-02-07 12:37:06	—
2020-02-10 11:00:01	iam_test_user_3	Yes	—	AKIA4EQBBG2YLHE37O6A	—	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_3	2020-02-07 12:37:06	—
2020-02-10 11:00:01	iam_test_user_4	Yes	—	AKIA4EQBBG2YAIACVUFX	myS3group	deny_all	—	arn:aws:iam::834299573936:user/iam_test_user_4	2020-02-07 12:37:06	—
2020-02-10 11:00:01	iam_test_user_5	Yes	—	AKIA4EQBBG2YNUSUX3GT	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_5	2020-02-07 12:37:07	—
2020-02-10 11:01:48	iam_test_user_6	No	—	AKIA4EQBBG2YHQAOWDWO	—	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_6	2020-02-07 12:37:07	—
2020-02-10 11:00:01	adminuser	Yes	2020-02-07 12:34:38	AKIA4EQBBG2YO3VWDSN6,AKIA4EQBBG2YFGH789IJ	system-admins	AmazonRoute53ReadOnlyAccess	Name,Email,Environment,Account_Type	arn:aws:iam::834299573936:user/adminuser	2019-10-31 16:23:07	Yes

Displaying 1 - 7 of 7


AWS IAM Users

Password last used	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
—	AKIA4EQBBG2YHCALR7UT,AKIA4EQBBG2YGLNUPO64	null_group,denyall_group	—	—	arn:aws:iam::834299573936:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPNPT	Active	···
—	AKIA4EQBBG2YCNNCIYP5	denyall_group	—	—	arn:aws:iam::834299573936:user/iam_test_user_2	2020-02-07 12:37:06	—	AIDA4EQBBG2YFLQ4677OF	Active	···
—	AKIA4EQBBG2YLHE37O6A	—	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_3	2020-02-07 12:37:06	—	AIDA4EQBBG2YOZ2ZX366W	Active	···
—	AKIA4EQBBG2YAIACVUFX	myS3group	deny_all	—	arn:aws:iam::834299573936:user/iam_test_user_4	2020-02-07 12:37:06	—	AIDA4EQBBG2YGGGAERS2G	Active	···
—	AKIA4EQBBG2YNUSUX3GT	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_5	2020-02-07 12:37:07	—	AIDA4EQBBG2YGJETIQNEQ	Active	···
—	AKIA4EQBBG2YHQAOWDWO	—	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_6	2020-02-07 12:37:07	—	AIDA4EQBBG2YPYJYCKNCT	Active	···
2020-02-07 12:34:38	AKIA4EQBBG2YO3VWDSN6,AKIA4EQBBG2YFGH789IJ	system-admins	AmazonRoute53ReadOnlyAccess	Name,Email,Environment,Account_Type	arn:aws:iam::834299573936:user/adminuser	2019-10-31 16:23:07	Yes	AIDA4EQBBG2YGZOQXT2JB	Active	···

Displaying 1 - 7 of 7

Note: If all unfiltered users are listed the default user for the integration will be indicated by "Yes" in the "Default user" field.

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_access_key_filter`	`text`	No	`-`	Filter users or access keys based on access keys applied to user. Filter by access key id, can be a string or regular expression.
`aws_iam_group_filter`	`text`	No	`-`	Filter users based on group membership. Filter by group name, can be a string or regular expression.

Name	Type	Required	Example	Tooltip
`aws_iam_policy_filter`	`text`	No	`-`	Filter users based on policies applied to user. Filter by policy name, can be a string or regular expression.
`aws_iam_query_type`	`select`	No	`-`	Type of query to perform for list_users, can be one of 'users' or 'access_keys'. Optional parameter.
`aws_iam_user_filter`	`text`	No	`-`	Filter users or access keys based on user name. Can be a string or regular expression.
`aws_iam_user_name`	`text`	No	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
            'version': '1.0', 'success': True, 'reason': None,
            'content': [{'Path': '/', 'UserName': 'iam_test_User',
'UserId': 'AIDA4EQBBG2YDOLTU6QSM',
                         'Arn':
'arn:aws:iam::123456789123:user/iam_test_User', 'CreateDate': '2019-11-05
15:54:43'},
                        {'Path': '/', 'UserName': 'iam_test_User_2',
'UserId': 'AIDA4EQBBG2YGZOQXT2JB',
                         'Arn':
'arn:aws:iam::123456789123:user/iam_test_User_2',
                         'CreateDate': '2019-10-31 16:23:07',
'PasswordLastUsed': '2019-11-12 10:55:42'}
                       ],
            'raw': '[{"Path": "/", "UserName": "iam_test_User", "UserId":
"AIDA4EQBBG2YDOLTU6QSM", "Arn":
"arn:aws:iam::834299573936:user/iam_test_User", "CreateDate": "2019-11-05
15:54:43"}, {"Path": "/", "UserName": "iam_test_User_2", "UserId":
"AIDA4EQBBG2YGZOQXT2JB", "Arn":
"arn:aws:iam::834299573936:user/iam_test_User_2", "CreateDate": "2019-10-
31 16:23:07"}]',
            'inputs': {},
            'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                        'host': 'myhost.ibm.com', 'execution_time_ms':
7951,
                        'timestamp': '2019-11-14 13:48:30'
```

```
                            }
}
```

▶ Example Pre-Process Script:

```
import re

# Get a list of all enabled filters.
ENABLED_FILTERS = [f for f in [rule.properties.aws_iam_user_filter,
rule.properties.aws_iam_group_filter,
                               rule.properties.aws_iam_policy_filter,
rule.properties.aws_iam_access_key_filter]
                   if f is not None]


def is_regex(regex_str):
    """"Test if sting is a correctly formed regular expression.

    :param regex_str: Regular expression string.
    :return: Boolean.
    """
    try:
        re.compile(regex_str)
        return True
    except re.error:
        return False


def main():
    # Test any enabled filters to ensure they are valid regular
expressions.
    for ef in (ENABLED_FILTERS):
        if not is_regex(ef):
            raise ValueError("The query filter '{}' is not a valid regular
expression.".format(unicode(ef)))

    inputs.aws_iam_user_filter = rule.properties.aws_iam_user_filter
    inputs.aws_iam_group_filter = rule.properties.aws_iam_group_filter
    inputs.aws_iam_policy_filter = rule.properties.aws_iam_policy_filter
    inputs.aws_iam_access_key_filter =
rule.properties.aws_iam_access_key_filter
    inputs.aws_iam_query_type = "users"


if __name__ == "__main__":
    main()
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_list_users script ##
#  Globals
import re
# List of fields in datatable fn_aws_iam_list_users script
DATA_TBL_FIELDS = ["query_execution_time", "UserName", "UserId", "Arn",
"DefaultUser", "CreateDate", "LoginProfileExists",
                   "PasswordLastUsed", "AccessKeyIds", "Policies", "Tags",
"Groups"]
FN_NAME = "fn_aws_iam_list_users"
WF_NAME = "List Users"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def check_add_quotes(tag_name):
    # Using regex
    # If spaces in tag name add quotes
    if re.search(r"\s", tag_name):
        return "'"+tag_name+"'"
    else:
        return tag_name

def process_access_key_ids(access_key_id_list, row):
    access_key_ids = []
    for ak_id in access_key_id_list:
        if ak_id["AccessKeyId"] is not None:
            access_key_ids.append(ak_id["AccessKeyId"])
    row.AccessKeyIds = ','.join(access_key_ids)

def process_policies(policy_list, row):
    policies = []
    for pol in policy_list:
        if pol["PolicyName"] is not None:
            policies.append(pol["PolicyName"])
    row.Policies = ','.join(policies)

def process_groups(group_list, row):
    groups = []
    for grp in group_list:
        if grp["GroupName"] is not None:
            groups.append(grp["GroupName"])
    row.Groups = ",".join(groups)

def process_tags(tag_list, row):
    tags = []
    for tag in tag_list:
        if tag["Key"] is not None:
            tags.append(tag["Key"])
    row.Tags = ','.join(check_add_quotes(t) for t in tags)

def main():
    note_text = ''
    filters = [f for f in [INPUTS["aws_iam_user_filter"],
INPUTS["aws_iam_group_filter"],
                           INPUTS["aws_iam_policy_filter"],
INPUTS["aws_iam_access_key_filter"]]
               if f is not None]
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>{1}</b> results returned for Resilient function " \
                    "<b>{2}</b>.".format(WF_NAME, len(CONTENT), FN_NAME)
        for u in CONTENT:
            newrow = incident.addRow("aws_iam_users")
            newrow.query_execution_date = QUERY_EXECUTION_DATE
            for f in DATA_TBL_FIELDS:
                newrow.Status = "Active"
                if u[f] is not None:
                    if isinstance(u[f], unicode) or isinstance(u[f], int)
\
                            or isinstance(u[f], long) or len(u[f]) == 0:
                        if f == "DefaultUser" and not u[f]:
                            pass
                        else:
                          newrow[f] = u[f]
                    else:
                        if f == "AccessKeyIds" and len(u[f]) > 0:
                            process_access_key_ids(u[f], newrow)
                        elif f == "Policies" and len(u[f]) > 0:
                            process_policies(u[f], newrow)
                        elif f == "Groups" and len(u[f]) > 0:
                            process_groups(u[f], newrow)
                        elif f == "Tags" and len(u[f]) > 0:
                            process_tags(u[f], newrow)
                        else:
                            newrow[f] = ','.join(u[f])
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>no</b> results returned for Resilient function <b>{1}</b>."\
            .format(WF_NAME, FN_NAME)
    if filters:
        note_text += "<br>Query Filters:</br>"
        if INPUTS.get("aws_iam_user_filter"):
            note_text += "<br>aws_iam_user_filter: <b>{0}</b>
</br>".format(INPUTS["aws_iam_user_filter"])
        if INPUTS.get("aws_iam_group_filter"):
            note_text += "<br>aws_iam_group_filter: <b>{0}</b>
</br>".format(INPUTS["aws_iam_group_filter"])
        if INPUTS.get("aws_iam_policy_filter"):
            note_text += "<br>aws_iam_policy_filter: <b>{0}</b>
</br>".format(INPUTS["aws_iam_policy_filter"])
        if INPUTS.get("aws_iam_access_key_filter"):
            note_text += "<br>aws_iam_access_key_filter: <b>{0}</b>
</br>".format(INPUTS["aws_iam_access_key_filter"])
    incident.addNote(helper.createRichText(note_text))
```

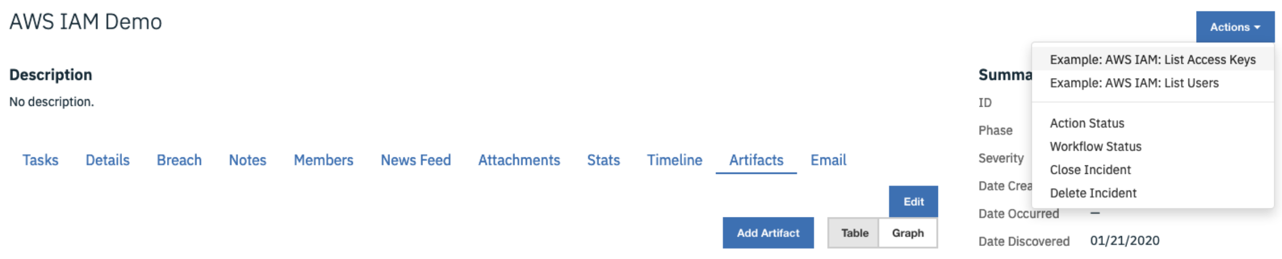
```
if __name__ == "__main__":
    main()
```

The workflow, `Example: AWS IAM: List Access keys`, sets the following input fields for the Function:

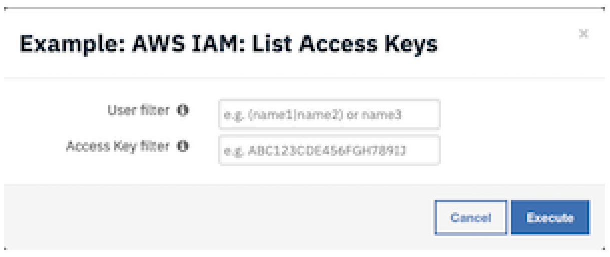
- aws_iam_user_filter (optional) is mapped to an activity field input. Note: Input should be a valid regular expression.
- aws_iam_access_key_filter (optional) is mapped to an activity field input. Note: Input should be a valid regular expression.
- aws_iam_query_type is set to value `access_keys`.

The workflow is initiated by the incident rule, `Example: AWS IAM: List Access keys`.

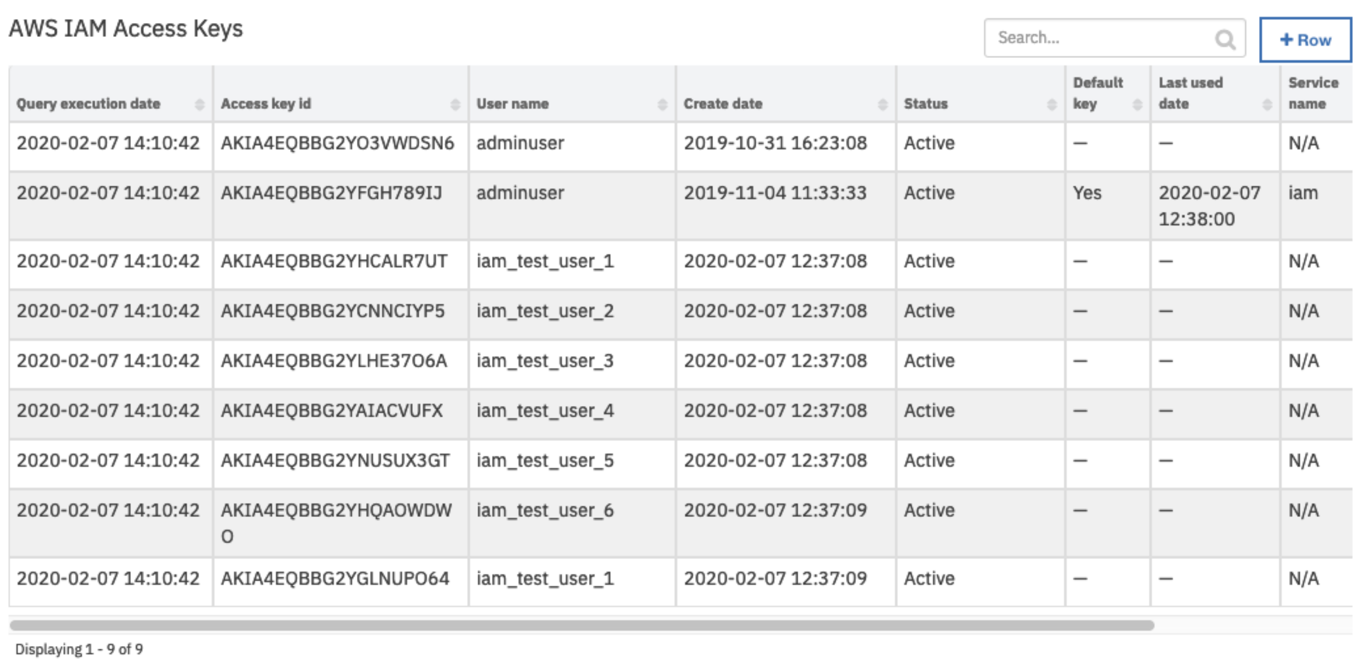
1. Open an incident and select `Example: AWS IAM: List Access keys` from Actions.



2. The user is presented with a list of input fields which can be used to filter users using regular expressions. Set any desired filters and click Execute..


Example: AWS IAM: List Access Keys
User filter
e.g. (name1|name2) or name3
Access Key filter
e.g. ABC123CDE456FGH789IJ
Cancel
Execute

This invokes the `Example: AWS IAM: List Access keys` workflow, which calls the `AWS IAM: List Users` function. The data table `AWS IAM Access Keys` will be updated in the Resilient platform with the users properties for the selected AWS account.



AWS IAM Access Keys

Search... + Row

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active	–	–	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YFGH789IJ	adminuser	2019-11-04 11:33:33	Active	Yes	2020-02-07 12:38:00	iam
2020-02-07 14:10:42	AKIA4EQBBG2YHCALR7UT	iam_test_user_1	2020-02-07 12:37:08	Active	–	–	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YCNNCIYP5	iam_test_user_2	2020-02-07 12:37:08	Active	–	–	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YLHE37O6A	iam_test_user_3	2020-02-07 12:37:08	Active	–	–	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YAIACVUFX	iam_test_user_4	2020-02-07 12:37:08	Active	–	–	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YNUSUX3GT	iam_test_user_5	2020-02-07 12:37:08	Active	–	–	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YHQAOWDWO	iam_test_user_6	2020-02-07 12:37:09	Active	–	–	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YGLNUPO64	iam_test_user_1	2020-02-07 12:37:09	Active	–	–	N/A

Displaying 1 - 9 of 9

Note: If all unfiltered access keys are listed the key for the default user for the integration will be indicated by "Yes" in the "Default key" field.

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_access_key_filter`	`text`	No	`-`	Filter users or access keys based on access keys applied to user. Filter by access key id, can be a string or regular expression.
`aws_iam_group_filter`	`text`	No	`-`	Filter users based on group membership. Filter by group name, can be a string or regular expression.
`aws_iam_policy_filter`	`text`	No	`-`	Filter users based on policies applied to user. Filter by policy name, can be a string or regular expression.
`aws_iam_query_type`	`select`	No	`-`	Type of query to perform for list_users, can be one of 'users' or 'access_keys'. Optional parameter.
`aws_iam_user_filter`	`text`	No	`-`	Filter users or access keys based on user name. Can be a string or regular expression.

Name	Type	Required	Example	Tooltip
`aws_iam_user_name`	`text`	No	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results =  {
            'version': '1.0', 'success': True, 'reason': None,
            'content': [{'Path': '/', 'UserName': 'iam_test_User',
'UserId': 'AIDA4EQBBG2YDOLTU6QSM',
                         'Arn':
'arn:aws:iam::123456789123:user/iam_test_User', 'CreateDate': '2019-11-05
15:54:43'},
                        {'Path': '/', 'UserName': 'iam_test_User_2',
'UserId': 'AIDA4EQBBG2YGZOQXT2JB',
                         'Arn':
'arn:aws:iam::123456789123:user/iam_test_User_2',
                         'CreateDate': '2019-10-31 16:23:07',
'PasswordLastUsed': '2019-11-12 10:55:42'}
                       ],
            'raw': '[{"Path": "/", "UserName": "iam_test_User", "UserId":
"AIDA4EQBBG2YDOLTU6QSM", "Arn":
"arn:aws:iam::834299573936:user/iam_test_User", "CreateDate": "2019-11-05
15:54:43"}, {"Path": "/", "UserName": "iam_test_User_2", "UserId":
"AIDA4EQBBG2YGZOQXT2JB", "Arn":
"arn:aws:iam::834299573936:user/iam_test_User_2", "CreateDate": "2019-10-
31 16:23:07"}]',
            'inputs': {},
            'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                        'host': 'myhost.ibm.com', 'execution_time_ms':
7951,
                        'timestamp': '2019-11-14 13:48:30'
                       }
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_access_key_filter =
rule.properties.aws_iam_access_key_filter
inputs.aws_iam_user_filter = rule.properties.aws_iam_user_filter
inputs.aws_iam_query_type = "access_keys"
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_list_users script ##
#  Globals
import re
# List of fields in datatable fn_aws_iam_list_users script main
DATA_TBL_FIELDS = ["query_execution_time", "UserName", "AccessKeyId",
"CreateDate", "Status", "DefaultKey"]
# List of fields in datatable fn_aws_iam_list_users script last used
access keys.
DATA_TBL_FIELDS_LUAK = ["LastUsedDate", "ServiceName", "Region"]
FN_NAME = "fn_aws_iam_list_users"
WF_NAME = "List Access Keys"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]

def process_access_keys(access_key_id_list, user_name):
    access_key_ids = []
    for ak_id in access_key_id_list:
        newrow = incident.addRow("aws_iam_access_keys")
        newrow.query_execution_date = QUERY_EXECUTION_DATE
        newrow.UserName = user_name
        for f in DATA_TBL_FIELDS[2:]:
            if ak_id[f] is not None:
                newrow[f] = ak_id[f]
        # Add key last used data if it exists.
        if ak_id["key_last_used"] is not None:
            luak = ak_id["key_last_used"]
            for l in DATA_TBL_FIELDS_LUAK:
                if luak[l] is not None:
                    newrow[l] = luak[l]
def main():
    note_text = ''
    filters = [f for f in [INPUTS["aws_iam_user_filter"],
INPUTS["aws_iam_group_filter"],
                           INPUTS["aws_iam_policy_filter"],
INPUTS["aws_iam_access_key_filter"]]
               if f is not None]
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>{1}</b> results returned for Resilient function " \
                   "<b>{2}</b>.".format(WF_NAME, len(CONTENT), FN_NAME)
        for u in CONTENT:
           if u["AccessKeyIds"]:
                user_name = u["UserName"]
                process_access_keys(u["AccessKeyIds"], user_name)

    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>no</b> results returned for Resilient function <b>{1}</b>."\
            .format(WF_NAME, FN_NAME)

    if filters:
        note_text += "<br>Query Filters:</br>"
        if INPUTS.get("aws_iam_user_filter"):
            note_text += "<br>aws_iam_user_filter: <b>{0}</b>
</br>".format(INPUTS["aws_iam_user_filter"])
        if INPUTS.get("aws_iam_access_key_filter"):
            note_text += "<br>aws_iam_access_key_filter: <b>{0}</b>
</br>".format(INPUTS["aws_iam_access_key_filter"])
    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Delete User

Delete the specified IAM user. Parameter aws_iam_user_name is an IAM user name.

When deleting an IAM user programmatically, the workflow will delete or remove the following items attached to the user:

```
Password ( DeleteLoginProfile )
Access keys ( DeleteAccessKey )
Inline policies ( DeleteUserPolicy )
Attached managed policies ( DetachUserPolicy )
Group memberships ( RemoveUserFromGroup )
```

Note: If any of other the following items is associated with the target user the integration will fail.

```
0Signing certificate ( DeleteSigningCertificate )
SSH public key ( DeleteSSHPublicKey )
Git credentials ( DeleteServiceSpecificCredential )
Multi-factor authentication (MFA) device ( DeactivateMFADevice ,
DeleteVirtualMFADevice )
```

- Example workflows that use this Resilient Function include `Example: AWS IAM: Delete User` or `Example: AWS IAM: Delete User For Artifact`.

Both of the example workflows are multi-step functions and will attempt to remove or delete the items referenced above if associated with the user, and will then attempt to delete the user.

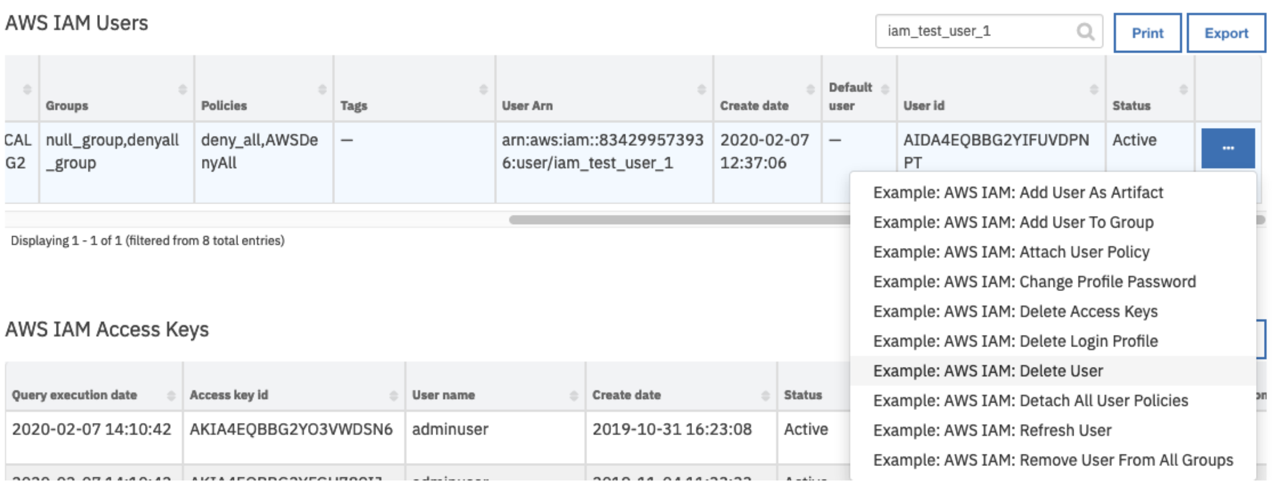
If any of the items mentioned above exist for the user the workflow will fail.

The workflow, `Example: AWS IAM: Delete User`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected data table row `Example: AWS IAM: Delete User` or artifact .

The workflow is initiated by the data table rule, `Example: AWS IAM: Delete User`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to the user whose access keys are to be deleted.
2. From the selected row's actions menu, select `Example: AWS IAM: Delete User`.


AWS IAM Users

iam_test_user_1 | Print | Export

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
CAL G2	null_group,denyall_group	deny_all,AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPNPT	Active	···

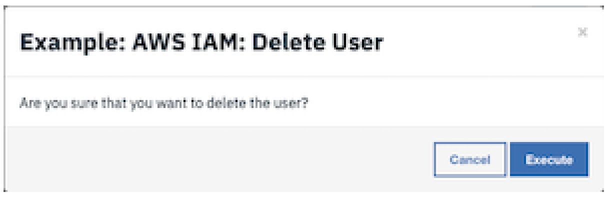
Displaying 1 - 1 of 1 (filtered from 8 total entries)

Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Detach All User Policies
Example: AWS IAM: Refresh User
Example: AWS IAM: Remove User From All Groups

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

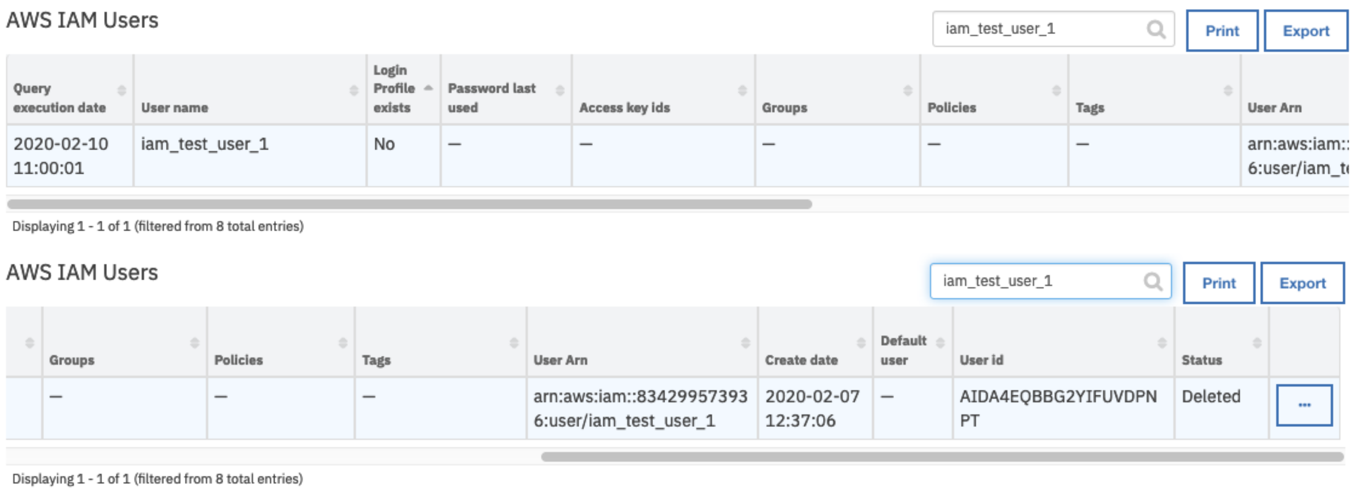
3. User is presented with a warning and an option to Execute or Cancel.


Example: AWS IAM: Delete User

Are you sure that you want to delete the user?

Pressing Execute invokes the `Example: AWS IAM: Delete User` workflow, which calls the `AWS IAM: Delete User` function.

On successful completion, the data table `AWS IAM Users` will be refreshed in the Resilient platform with the updated access key details for the selected user. The data table `Status` field will transition to `Deleted`.


AWS IAM Users

iam_test_user_1 | Print | Export

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:00:01	iam_test_user_1	No	—	—	—	—	—	arn:aws:iam::6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

AWS IAM Users

iam_test_user_1 | Print | Export

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPNPT	Deleted	···

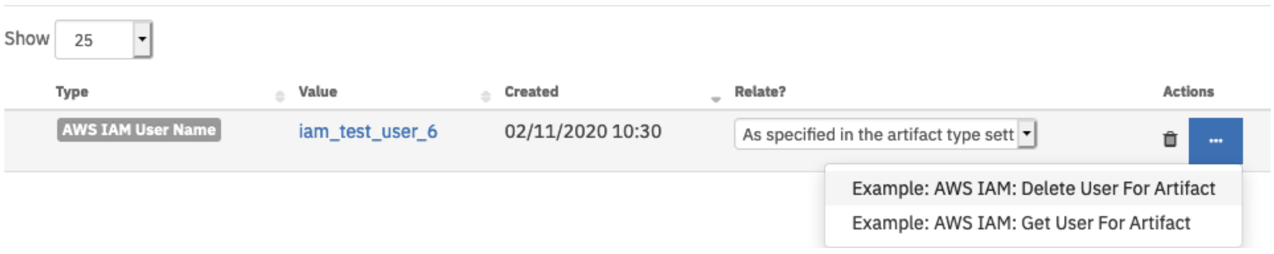
Displaying 1 - 1 of 1 (filtered from 8 total entries)

The workflow, `Example: AWS IAM: Delete User For Artifact`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to an artifact value for artifact of type `AWS IAM User Name`.

The workflow is initiated by the artifact rule, `Example: AWS IAM: Delete User For Artifact`.

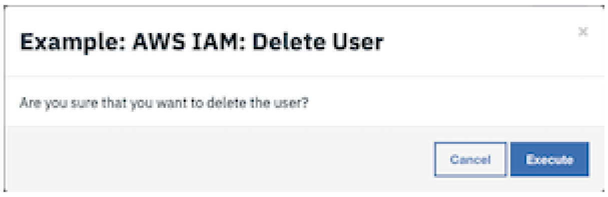
1. Open an incident and select the 'Artifacts' tab.
2. For a Resilient artifact of type, 'AWS IAM User Name' click Action-> `Example: AWS IAM: Delete User For Artifact`.



Type	Value	Created	Relate?	Actions
AWS IAM User Name	iam_test_user_6	02/11/2020 10:30	As specified in the artifact type sett	

Show 25
Example: AWS IAM: Delete User For Artifact
Example: AWS IAM: Get User For Artifact

3. User is presented with a warning and an option to Execute or Cancel.

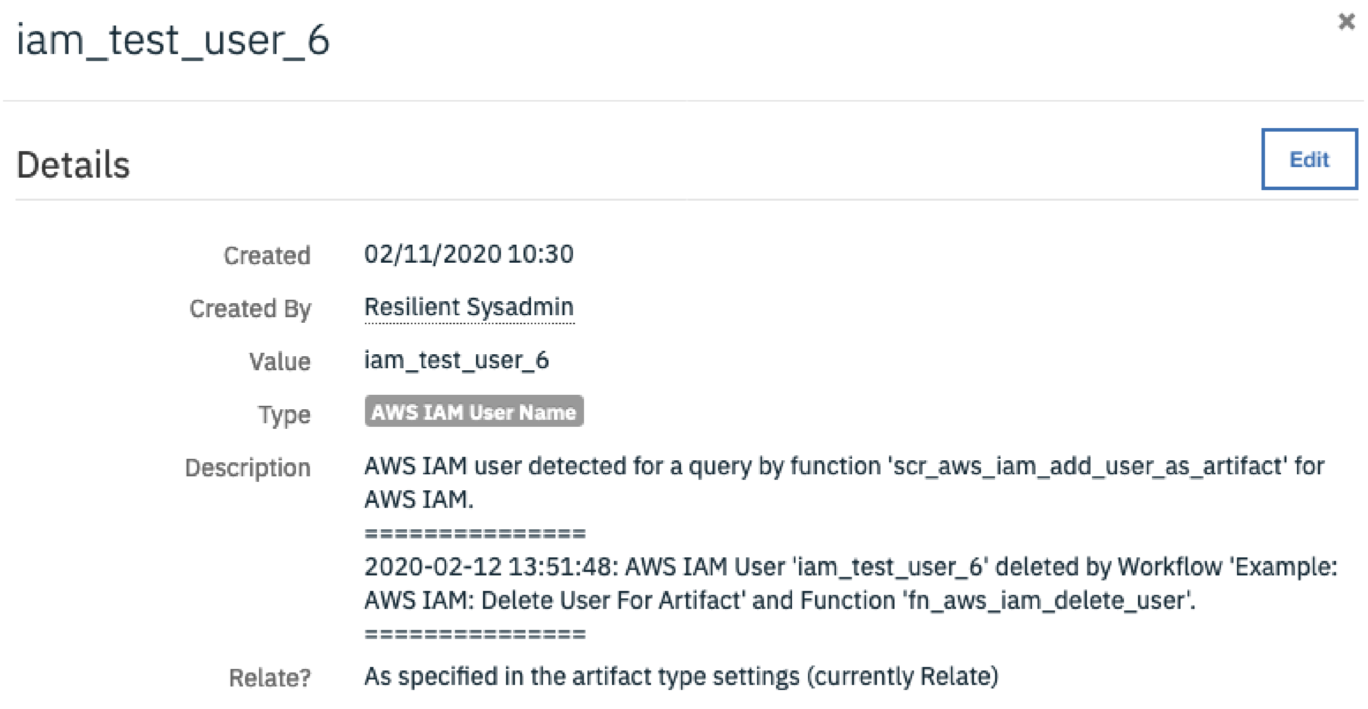


Example: AWS IAM: Delete User

Are you sure that you want to delete the user?

Pressing Execute invokes the `Example: AWS IAM: Delete User For Artifact` workflow, which calls the `AWS IAM: Delete User` function.

On successful completion, the artifact description will be updated with details of user deletion.



iam_test_user_6

Details

Created	02/11/2020 10:30
Created By	Resilient Sysadmin
Value	iam_test_user_6
Type	AWS IAM User Name
Description	AWS IAM user detected for a query by function 'scr_aws_iam_add_user_as_artifact' for AWS IAM. =============== 2020-02-12 13:51:48: AWS IAM User 'iam_test_user_6' deleted by Workflow 'Example: AWS IAM: Delete User For Artifact' and Function 'fn_aws_iam_delete_user'. ===============
Relate?	As specified in the artifact type settings (currently Relate)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this
code block.
    # To see view the output of a Function, run resilient-circuits in
DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_delete_access_keys script ##
# Example result:
"""
OK
Result: { 'version': '1.0', 'success': True, 'reason': None,
          'content': 'OK',
          'raw': '"OK"',
          'inputs': {'aws_iam_user_name': 'iam_test_user'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0', 'host': 'myhost.ibm.com',
                      'execution_time_ms': 689, 'timestamp': '2020-01-15
10:27:48'
                     }
}
"""
#  Globals
# List of fields in datatable for fn_aws_iam_delete_user  script
DATA_TBL_FIELDS = ["Status"]
FN_NAME = "fn_aws_iam_delete_user"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs

def main():
    note_text = ''
    if CONTENT:
        if CONTENT == "OK":
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: User
<b>{1}</b> was successfully deleted for " \
                        "Resilient function <b>{2}</b>.".format(WF_NAME,
INPUTS["aws_iam_user_name"], FN_NAME)
            row.Status = "Deleted"
            row.Tags = ''
        else:
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: 
Unexpected delete status <b>{1}</b> for delete" \
                        " user operation <b>{2}</b> for Resilient function 
<b>{3}</b>."\
                .format(WF_NAME, CONTENT, INPUTS["aws_iam_user_name"], 
FN_NAME)
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was 
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Delete Access Keys

Delete the access key pairs associated with the specified IAM user. Parameter aws_iam_user_name is an IAM user name. Parameter aws_iam_access_keys is a comma separated list of IAM access key ids.

- Example workflows that use this Resilient Function include `Example: AWS IAM: Delete Access Keys`, `Example: AWS IAM: Delete Access Key`, `Example: AWS IAM: Delete Access Key For Artifact`, `Example: AWS IAM: Delete User` or `Example: AWS IAM: Delete User For Artifact`.

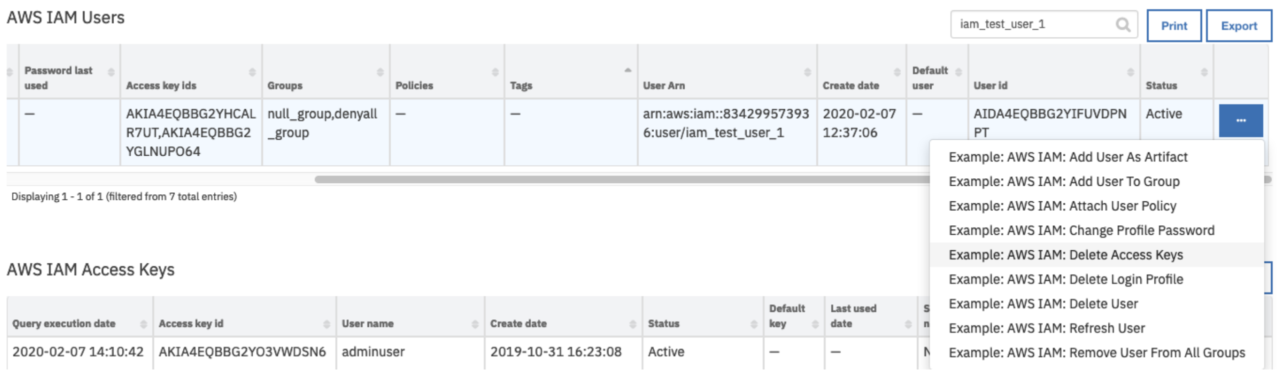
The workflow, `Example: AWS IAM: Delete Access Keys`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.
- aws_iam_access_keys is mapped to all access keys for the user from the selected row of data table `AWS IAM Users`.

The workflow is initiated by the data table rule, `Example: AWS IAM: Delete Access Keys`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to the user whose access keys are to be deleted.

2. From the selected row's actions menu, select `Example: AWS IAM: Delete Access Keys`.



AWS IAM Users

iam_test_user_1

Password last used	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status
—	AKIA4EQBBG2YHCALR7UT,AKIA4EQBBG2YGLNUPO64	null_group,denyall_group	—	—	arn:aws:iam::834299573936:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPNPT	Active

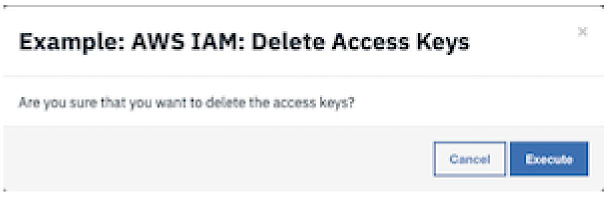
Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—

Print
Export
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Refresh User
Example: AWS IAM: Remove User From All Groups

3. The user is presented with a warning and an option to Execute or Cancel.

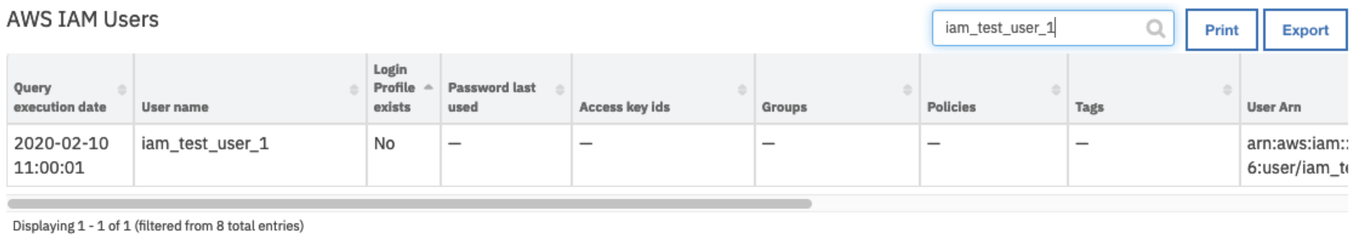


Example: AWS IAM: Delete Access Keys

Are you sure that you want to delete the access keys?

Pressing Execute invokes the `Example: AWS IAM: Delete Access Keys` workflow, which calls the `AWS IAM: Delete Access Keys` function.

On successful completion, for the data table `AWS IAM Users` the `Access key ids` field will get updated for the selected user.



AWS IAM Users

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:00:01	iam_test_user_1	No	–	–	–	–	–	arn:aws:iam:: 6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

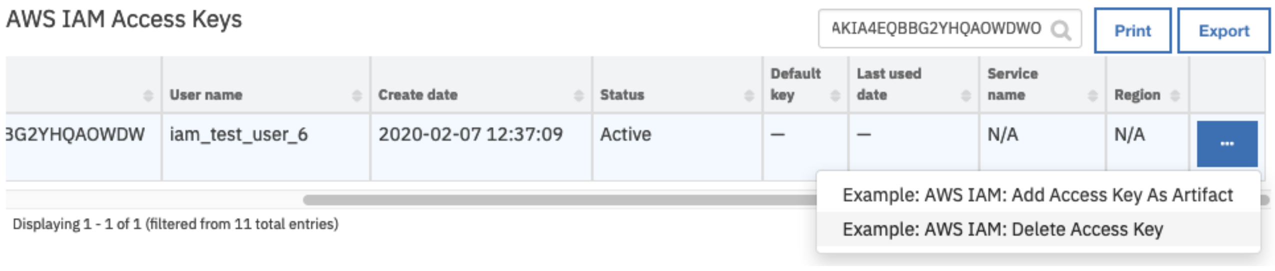
The workflow, `Example: AWS IAM: Delete Access Key`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Access Keys`.
- aws_iam_access_keys is mapped to the access key id from the selected row of data table `AWS IAM Access Keys`.

The workflow is initiated by the data table rule, `Example: AWS IAM: Delete Access Key`.

1. Open an incident and select the row of data table `AWS IAM Access Keys` corresponding to the access key to be deleted.

2. From the selected row's actions menu, select `Example: AWS IAM: Delete Access Key`.



AWS IAM Access Keys

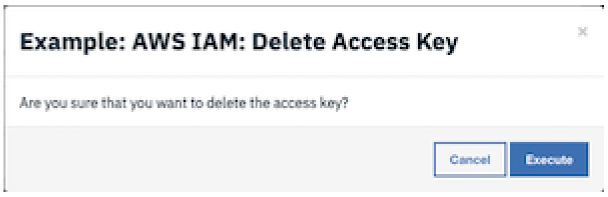
AKIA4EQBBG2YHQAOWDWO
Print
Export

	User name	Create date	Status	Default key	Last used date	Service name	Region	
3G2YHQAOWDW	iam_test_user_6	2020-02-07 12:37:09	Active	–	–	N/A	N/A	...

Displaying 1 - 1 of 1 (filtered from 11 total entries)

Example: AWS IAM: Add Access Key As Artifact
Example: AWS IAM: Delete Access Key

3. User is presented with a warning and an option to Execute or Cancel.

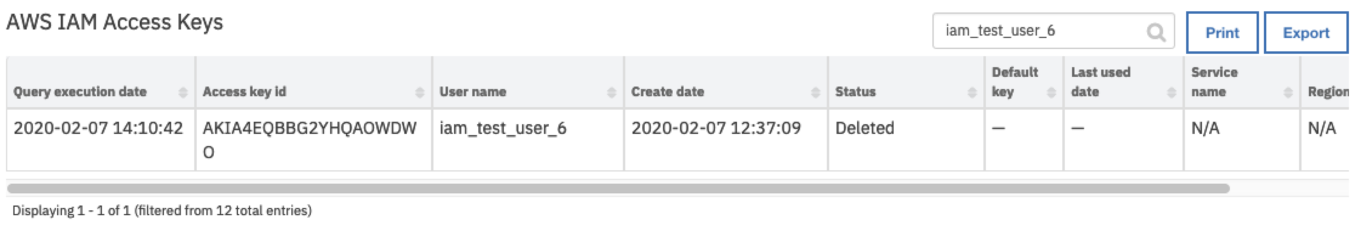


Example: AWS IAM: Delete Access Key

Are you sure that you want to delete the access key?

Pressing Execute invokes the `Example: AWS IAM: Delete Access Keys` workflow, which calls the `AWS IAM: Delete Access Keys` function.

On successful completion, the data table `AWS IAM Access Keys` `Status` field will transition to `Deleted`.


AWS IAM Access Keys

iam_test_user_6

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name	Region
2020-02-07 14:10:42	AKIA4EQBBG2YHQAOWDWO	iam_test_user_6	2020-02-07 12:37:09	Deleted	—	—	N/A	N/A

Displaying 1 - 1 of 1 (filtered from 12 total entries)

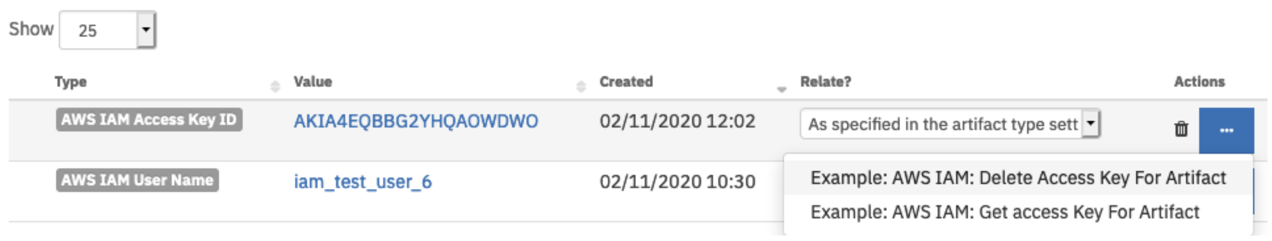
The workflow, `Example: AWS IAM: Delete Access Key For Artifact`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name computed from a previous step in the workflow.
- aws_iam_access_keys is mapped to an artifact value for artifact of type `AWS IAM Access Key ID`.

The workflow is initiated by the artifact rule, `Example: AWS IAM: Delete Access Key For Artifact`.

1. Open an incident and select the 'Artifacts' tab.

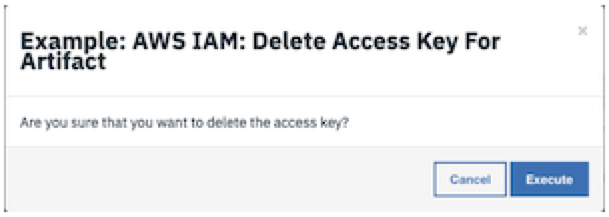
2. For a Resilient artifact of type, `AWS IAM Access Key ID` click Action-> `Example: AWS IAM: Delete Access Key For Artifact`.


Show 25

Type	Value	Created	Relate?	Actions
AWS IAM Access Key ID	AKIA4EQBBG2YHQAOWDWO	02/11/2020 12:02	As specified in the artifact type sett	
AWS IAM User Name	iam_test_user_6	02/11/2020 10:30		

Example: AWS IAM: Delete Access Key For Artifact
Example: AWS IAM: Get access Key For Artifact

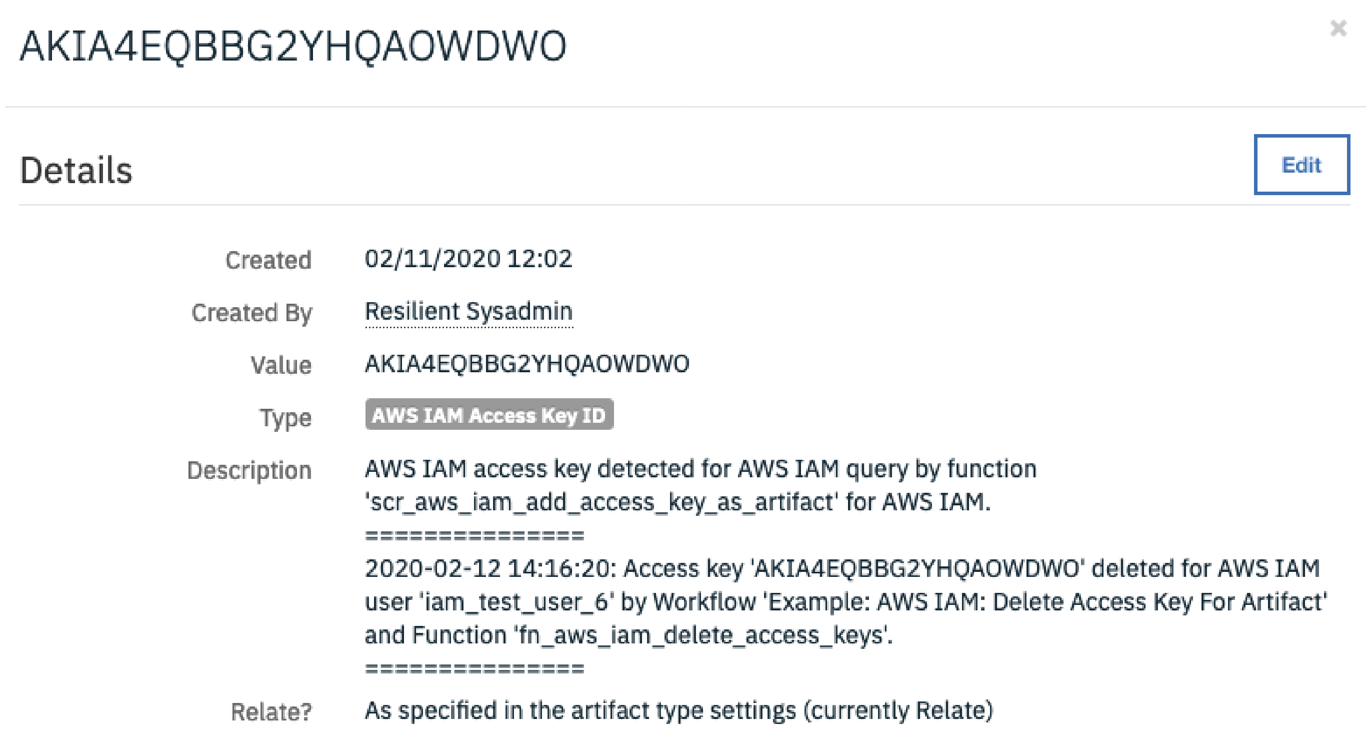
3. User is presented with a warning and an option to Execute or Cancel.


Example: AWS IAM: Delete Access Key For Artifact

Are you sure that you want to delete the access key?

Pressing Execute invokes the `Example: AWS IAM: Delete Access Key For Artifact` workflow, which calls the `AWS IAM: Delete Access Keys` function.

On successful completion, the artifact description will be updated with details of access key deletion.



AKIA4EQBBG2YHQAOWDWO

Details

Edit

Created	02/11/2020 12:02
Created By	Resilient Sysadmin
Value	AKIA4EQBBG2YHQAOWDWO
Type	AWS IAM Access Key ID
Description	AWS IAM access key detected for AWS IAM query by function 'scr_aws_iam_add_access_key_as_artifact' for AWS IAM. =============== 2020-02-12 14:16:20: Access key 'AKIA4EQBBG2YHQAOWDWO' deleted for AWS IAM user 'iam_test_user_6' by Workflow 'Example: AWS IAM: Delete Access Key For Artifact' and Function 'fn_aws_iam_delete_access_keys'. ===============
Relate?	As specified in the artifact type settings (currently Relate)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_access_keys`	`text`	Yes	`-`	Comma seperated list of AWS IAM access key names.
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this code block.
    # To see view the output of a Function, run resilient-circuits in DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
inputs.aws_iam_access_keys = row.AccessKeyIds
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_delete_access_keys script ##
# Example result:
"""
OK
Result: {
          'version': '1.0', 'success': True, 'reason': None,
          'content': [{'AccessKeyId': 'AKIA4EQBBG2YKXYJB55L', 'Status':
'OK'},
                      {'AccessKeyId': 'AKIA4EQBBG2YKXYJB55M', 'Status':
'NoSuchEntity'}],
          'raw': '[{"AccessKeyId": "AKIA4EQBBG2YKXYJB55L", "Status":
"OK"},
                  {"AccessKeyId": "AKIA4EQBBG2YKXYJB55M", "Status":
"NoSuchEntity"}]',
          'inputs': {'aws_iam_user_name': 'iam_johnpren_test_User',
'aws_iam_access_keys': 'AKIA4EQBBG2YKXYJB55L,AKIA4EQBBG2YKXYJB55M'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                      'host': 'myhost.ibm.com', 'execution_time_ms':
37199, 'timestamp': '2019-11-21 14:31:13'
                     }
}
"""
#  Globals
# List of fields in datatable for fn_aws_iam_delete_access_keys  script
DATA_TBL_FIELDS = ["AccessKeyIds"]
FN_NAME = "fn_aws_iam_delete_access_keys"
WF_NAME = "Delete Access Keys"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    deleted = 0
    no_such_entity = 0
    deleted_keys = []
    no_such_entity_keys = []
    if CONTENT:
        for ak_stat in CONTENT:
            if ak_stat["Status"] == "OK":
                deleted += 1
                deleted_keys.append(ak_stat["AccessKeyId"])
            else:
                no_such_entity += 1
                no_such_entity_keys.append(ak_stat["AccessKeyId"])
        if deleted_keys:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> Access Key Ids <b>{2}</b> deleted " \
                        "for user <b>{3}</b> for Resilient function <b>{4}
</b>."\
                .format(WF_NAME, len(deleted_keys), ", ".join(str(i) for i
in deleted_keys), INPUTS["aws_iam_user_name"], FN_NAME)
        if no_such_entity:
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: There
were <b>{1}</b> Access Key Ids <b>{2}</b> " \
                        "which did not exist for user <b>{3}</b> for
Resilient function <b>{4}</b>."\
                .format(WF_NAME, len(no_such_entity_keys), ",
".join(str(i) for i in no_such_entity_keys), INPUTS["aws_iam_user_name"],
FN_NAME)
        row.AccessKeyIds = ""
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Remove User From Groups

Removes the specified IAM user from the specified groups. Group names is be a comma separated string of group names. Parameter aws_iam_user_name is an IAM user name. Parameter aws_iam_group_names is a comma separated list of IAM group names.

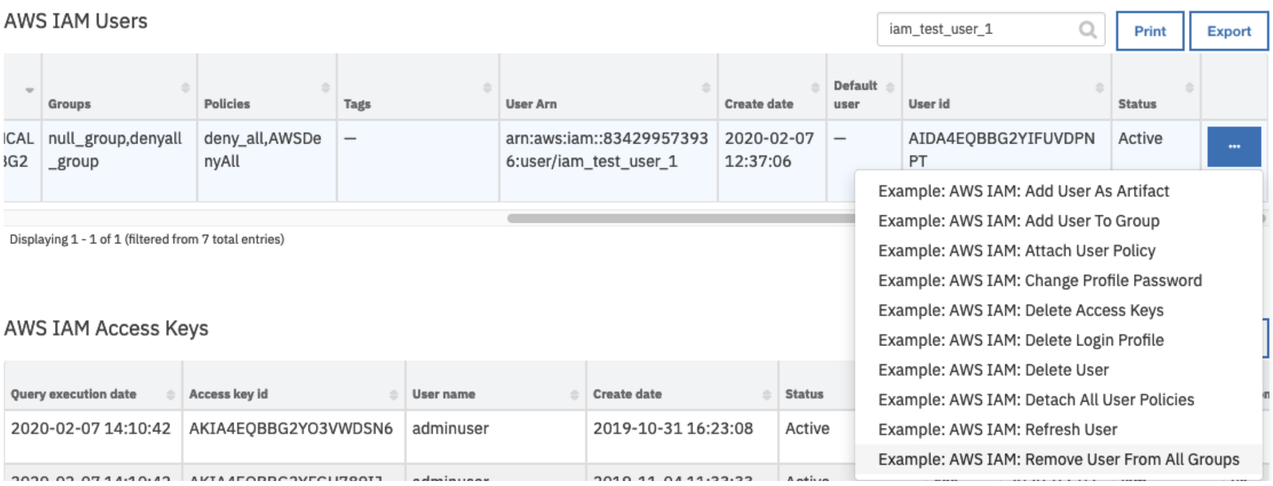
- Example workflows that use this Resilient Function include `Example: AWS IAM: Remove User From All Groups`, `Example: AWS IAM: Delete User` or `Example: AWS IAM: Delete User For Artifact`.

The workflow, `Example: AWS IAM: Remove User From All Groups`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.
- aws_iam_group_names is mapped to all group names from the selected row of data table `AWS IAM Users`.

The workflow is initiated by the data table rule, `Example: AWS IAM: Remove User From All Groups`.

1. Open an incident and select the row of data table `AWS IAM Users`corresponding to user who needs to be removed from all groups.
2. From the selected row's actions menu, select `Example: AWS IAM: Remove User From All Groups`.



AWS IAM Users

iam_test_user_1 Print Export

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
ICAL 3G2	null_group,denyall_group	deny_all,AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPNPT	Active	...

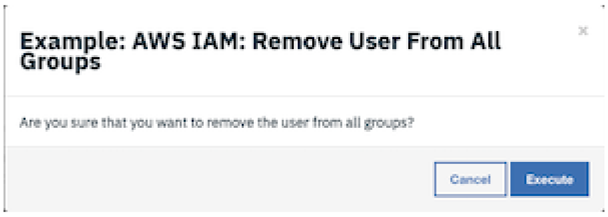
Displaying 1 - 1 of 1 (filtered from 7 total entries)

Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Detach All User Policies
Example: AWS IAM: Refresh User
Example: AWS IAM: Remove User From All Groups

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

3. User is presented with a warning and an option to Execute or Cancel.

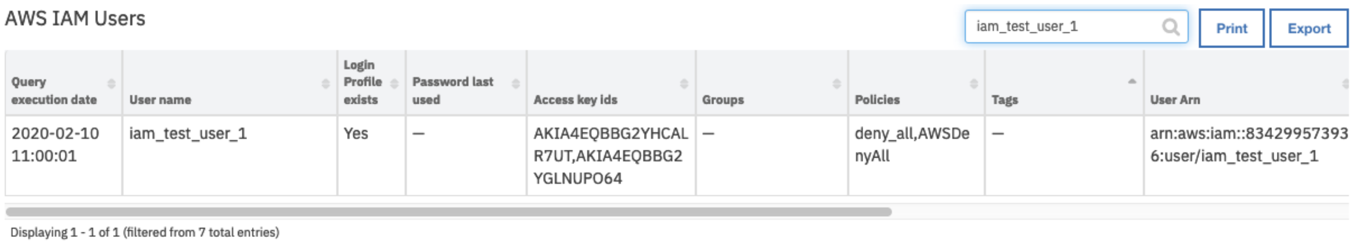


Example: AWS IAM: Remove User From All Groups

Are you sure that you want to remove the user from all groups?

Pressing Execute invokes the `Example: AWS IAM: Remove User From All Groups` workflow, which calls the `AWS IAM: Remove User From Groups` function.

On successful completion, for the data table `AWS IAM Users` the `Groups` field will get updated to an empty value for the selected user.



AWS IAM Users

iam_test_user_1 Print Export

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:00:01	iam_test_user_1	Yes	—	AKIA4EQBBG2YHCALR7UT,AKIA4EQBBG2YGLNUPO64	—	deny_all,AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_1

Displaying 1 - 1 of 1 (filtered from 7 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_group_names`	`text`	Yes	`-`	Comma separated list of AWS IAM group names.
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this
code block.
    # To see view the output of a Function, run resilient-circuits in
DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
inputs.aws_iam_group_names = row.Groups
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_detach_user_policies script ##
# Example result:
"""
OK
Result: {
          'version': '1.0', 'success': True, 'reason': None,
          'content': [{'PolicyArn': 'arn:aws:iam::aws:policy/AWSDenyAll',
'Status': 'OK'}
                      {'PolicyArn':
'arn:aws:iam::aws:policy/AWSDenyAll_2', 'Status': 'NoSuchEntity'}],
          'raw': '[{'PolicyArn': "arn:aws:iam::aws:policy/AWSDenyAll",
'Status": 'OK'},
                  {'PolicyArn': 'arn:aws:iam::aws:policy/AWSDenyAll_2',
'Status': 'NoSuchEntity'}]',
          'inputs': {'aws_iam_arns': 'arn:aws:iam::aws:policy/AWSDenyAll',
'aws_iam_user_name': 'iam_test_User_1'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0', 'host': 'myhost.ibm.com',
                      'execution_time_ms': 790, 'timestamp': '2019-11-29
12:18:30'
                     }
}
"""
#  Globals
# List of fields in datatable for fn_aws_iam_detach_user_policies  script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_remove_user_from_groups"
WF_NAME = "Remove User From All Groups"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    added = 0
    no_such_entity = 0
    added_groups = []
    no_such_entity_groups = []
    if CONTENT:
        for pol_stat in CONTENT:
            if pol_stat["Status"] == "OK":
                added += 1
                added_groups.append(pol_stat["GroupName"])
            else:
                no_such_entity += 1
                no_such_entity_groups.append(pol_stat["GroupName"])
        if added_groups:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> Groups <b>{2}</b> removed " \
                        "for user <b>{3}</b> for Resilient function <b>{4}
</b>."\
                .format(WF_NAME, len(added_groups), ", ".join(str(i) for i
in added_groups), INPUTS["aws_iam_user_name"], FN_NAME)
        if no_such_entity:
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: There
were <b>{1}</b> Groups <b>{2}</b> " \
                        "which did not exist for user <b>{3}</b> for
Resilient function <b>{4}</b>."\
                .format(WF_NAME, len(no_such_entity_groups), ",
".join(str(i) for i in no_such_entity_groups),
INPUTS["aws_iam_user_name"], FN_NAME)
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Attach User policies

Attach the specified managed policies to the specified IAM user. Parameter aws_iam_user_name is an IAM user name. Parameter aws_iam_policy_names (optional) is a comma separated list of IAM policy names. Parameter (optional) aws_iam_arns is a comma separated list of IAM policy arns.

Note: One of parameters aws_iam_policy_names or aws_iam_arns required to be set.

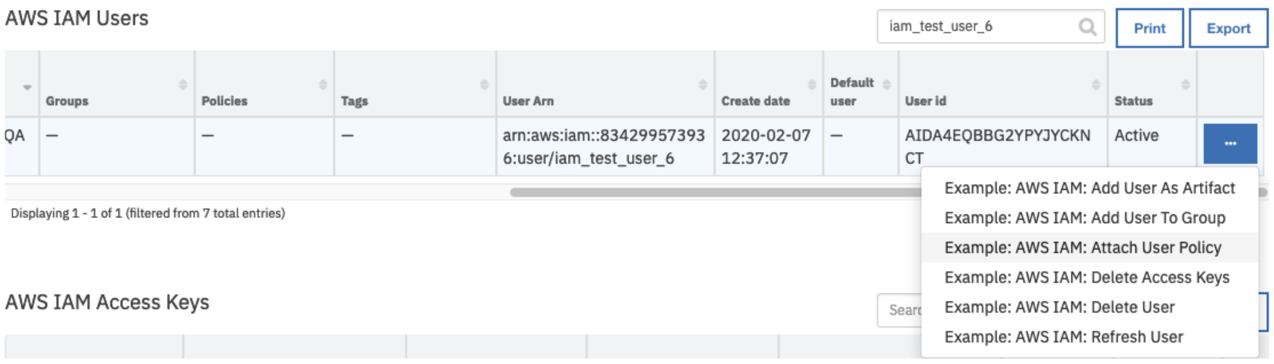
- Example workflows that use this Resilient Function include `Example: AWS IAM: Attach User Policy`

The workflow, `Example: AWS IAM: Attach User Policy`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.
- aws_iam_policy_names is mapped to activity field `aws_iam_policy_name` which should be a drop-down list of policy names.

The workflow is initiated by the data table rule, `Example: AWS IAM: Attach User Policy`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to user who needs to have a policy attached.
2. From the selected row's actions menu, select `Example: AWS IAM: Attach User Policy`.


AWS IAM Users

iam_test_user_6 Print Export

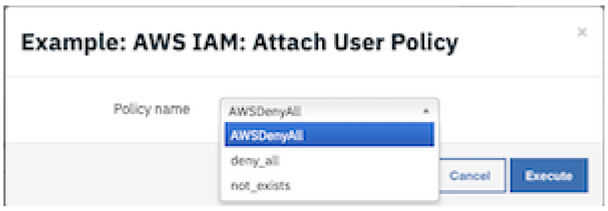
	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
QA	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_6	2020-02-07 12:37:07	—	AIDA4EQBBG2YPYJYCKNCT	Active	···

Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete User
Example: AWS IAM: Refresh User

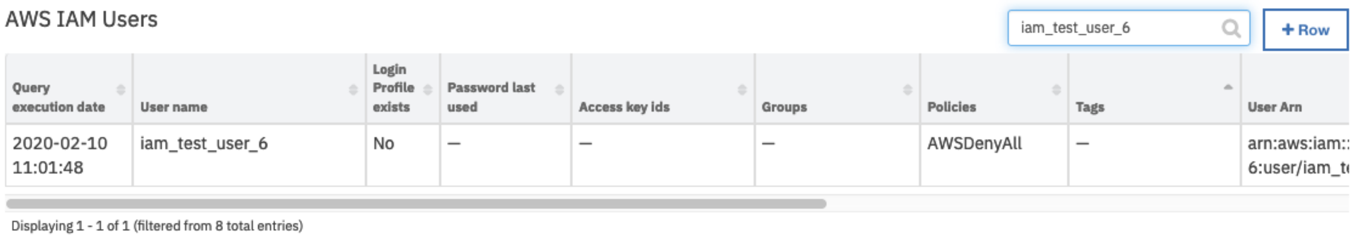
3. From the drop-down list of user defined polciy name , select a policy and click Execute.


Example: AWS IAM: Attach User Policy

Policy name: AWSDenyAll

This invokes the `Example: AWS IAM: Attach User Policy` workflow, which calls the `AWS IAM: Attach User policies` function.

On successful completion, for the data table `AWS IAM Users` the `Policies` field will get updated for the selected user.


AWS IAM Users

iam_test_user_6 + Row

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:01:48	iam_test_user_6	No	—	—	—	AWSDenyAll	—	arn:aws:iam:: 6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_arns`	`text`	No	`-`	Comma separated list of AWS IAM Amazon Resource Names (ARNs).
`aws_iam_policy_names`	`text`	No	`-`	Comma separated list of AWS IAM policy names.
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

Note: At least One of the parameters `aws_iam_arns` or `aws_iam_user_name` must be set.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this
code block.
    # To see view the output of a Function, run resilient-circuits in
DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
inputs.aws_iam_policy_names = rule.properties.aws_iam_policy_name
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_attach_user_policies script ##
# Example result:
"""
OK
Result: {
          'version': '1.0', 'success': True, 'reason': None,
          'content': [{'PolicyArn': 'arn:aws:iam::aws:policy/AWSDenyAll',
'Status': 'OK'},
                      {'PolicyArn':
'arn:aws:iam::aws:policy/AWSDenyAll_2', 'Status': 'NoSuchEntity'}],
          'raw': '[{'PolicyArn': "arn:aws:iam::aws:policy/AWSDenyAll",
'Status": 'OK'},
                  {'PolicyArn': 'arn:aws:iam::aws:policy/AWSDenyAll_2',
'Status': 'NoSuchEntity'}]',
          'inputs': {'aws_iam_arns': 'arn:aws:iam::aws:policy/AWSDenyAll',
'aws_iam_user_name': 'iam_test_User_1'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0', 'host': 'myhost.ibm.com',
                      'execution_time_ms': 790, 'timestamp': '2019-11-29
12:18:30'
                     }
}
"""
#  Globals
# List of fields in datatable for fn_aws_iam_attach_user_policies  script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_attach_user_policies"
WF_NAME = "Attach User Policy"
# Processing
CONTENT = results.content
```

```
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    added = 0
    no_such_entity = 0
    added_policies = []
    no_such_entity_policies = []
    if CONTENT:
        for pol_stat in CONTENT:
            if pol_stat["Status"] == "OK":
                added += 1
                added_policies.append(pol_stat["PolicyName"])
            else:
                no_such_entity += 1
                no_such_entity_policies.append(pol_stat["PolicyName"])
        if added_policies:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> Policies <b>{2}</b> added " \
                        "for user <b>{3}</b> for Resilient function <b>{4}
</b>."\
                .format(WF_NAME, len(added_policies), ", ".join(str(i) for
i in added_policies), INPUTS["aws_iam_user_name"], FN_NAME)
        if no_such_entity:
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: There
were <b>{1}</b> Policies <b>{2}</b> " \
                        "which did not exist for user <b>{3}</b> for
Resilient function <b>{4}</b>."\
                .format(WF_NAME, len(no_such_entity_policies), ",
".join(str(i) for i in no_such_entity_policies),
INPUTS["aws_iam_user_name"], FN_NAME)
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Update Login Profile

Change the password for the specified IAM user. Parameter aws_iam_user_name is an IAM user name. Parameter aws_iam_password is a new password value for an IAM user. Parameter aws_iam_password_reset_required is a boolean value to determine whether a password reset should be required on change.

- Example workflows that use this Resilient Function include `Example: AWS IAM: Change Profile Password`

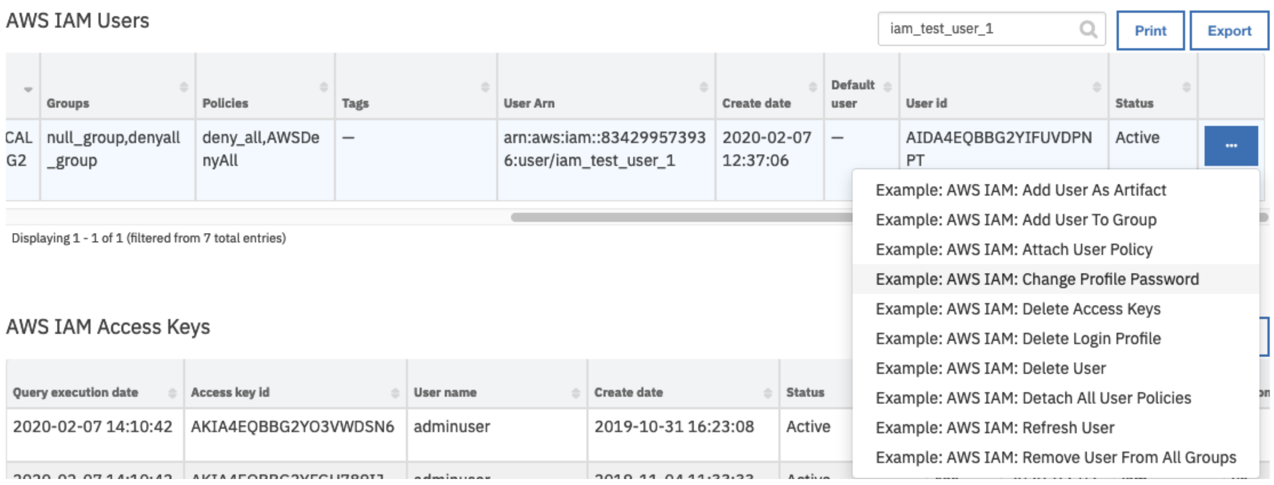
The workflow, `Example: AWS IAM: Change Profile Password`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.
- aws_iam_password is mapped to an activity field input.
- aws_iam_password_reset_required is mapped to a boolean from an activity field drop-down list.

The workflow is initiated by the data table rule, `Example: AWS IAM: Change Profile Password`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to the user who needs to have a profile password updated.

2. From the selected row's actions menu, select `Example: AWS IAM: Change Profile Password`.



AWS IAM Users

iam_test_user_1

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
CAL G2	null_group,denyall _group	deny_all,AWSDe nyAll	—	arn:aws:iam::83429957393 6:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPN PT	Active	•••

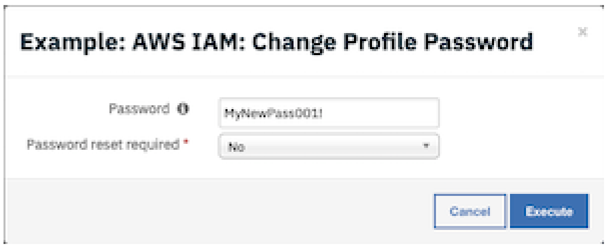
Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

Print
Export
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Detach All User Policies
Example: AWS IAM: Refresh User
Example: AWS IAM: Remove User From All Groups

3. The user is presented with 2 activity fields for a new password a password confirmation and a boolean to indicate if password reset is needed. Set the appropriate values for the fields and click Execute.



Example: AWS IAM: Change Profile Password

Password	MyNewPass001!
Password reset required *	No

Note: The minimum reqirements for a new password is at > 8 characters, at least 1 uppercase and 1 lowercase ascii character.

This invokes the `Example: AWS IAM: Change Profile Password` workflow, which calls the `AWS IAM: Update Login Profile` function.

On successful completion a note will be created indicting status of action.

▶ Inputs:

Name	Type	Required	Example	Tooltip

Name	Type	Required	Example	Tooltip
`aws_iam_password`	`text`	Yes	`-`	AWS IAM password for user login profile.
`aws_iam_password_reset_required`	`boolean`	Yes	`-`	A password reset required on password change.
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this code block.
    # To see view the output of a Function, run resilient-circuits in DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
# Test password to see it complies with basic password policy.
err_msg = "The new password needs be minimum 8 characters in length and have at least 1 uppercase and 1 lowercase character."
if len(rule.properties.aws_iam_password) < 8:
    raise ValueError(err_msg)
if not any(c.isupper() for c in rule.properties.aws_iam_password):
    raise ValueError(err_msg)
if not any(c.islower() for c in rule.properties.aws_iam_password):
    raise ValueError(err_msg)
inputs.aws_iam_password = rule.properties.aws_iam_password
inputs.aws_iam_password_reset_required = rule.properties.aws_iam_password_reset_required
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_delete_login_profile script ##
# Example result:
"""
OK
Result: {
         'version': '1.0', 'success': True, 'reason': None,
         'content': 'NoSuchEntity', 'raw': '"NoSuchEntity"',
         'inputs': {'aws_iam_user_name': 'iam_test_User'},
         'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                     'host': 'myhost.ie.ibm.com', 'execution_time_ms':
9170, 'timestamp': '2019-11-18 16:24:17'
                     }
}
NosuchEntity
Result: {
         'version': '1.0', 'success': True, 'reason': None,
         'content': 'OK', 'raw': '"OK"',
         'inputs': {'aws_iam_user_name': 'iam_test_User'},
         'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                     'host': 'myhost.ie.ibm.com', 'execution_time_ms':
9170, 'timestamp': '2019-11-18 16:24:17'
                     }
}
"""
#  Globals
# List of fields in datatable fn_aws_iam_delete_login_profile  script
DATA_TBL_FIELDS = ["Groups"]
FN_NAME = "fn_aws_iam_update_login_profile"
WF_NAME = "Update Login Profile"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    if CONTENT:
        if CONTENT == "OK":
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: Login
profile updated for user <b>{1}</b> for " \
                        "Resilient function <b>{2}</b>.".format(WF_NAME,
INPUTS["aws_iam_user_name"], FN_NAME)
        elif CONTENT == "PasswordPolicyViolation":
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>:
Password policy violation updating user <b>{1}</b> for " \
                        "Resilient function <b>{2}</b>.".format(WF_NAME,
INPUTS["aws_iam_user_name"], FN_NAME)
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Add User To Groups

Add the specified IAM user to the specified groups. Parameter aws_iam_user_name is an IAM user name. Parameter aws_iam_group_names is a comma separated list of IAM group names.

- Example workflows that use this Resilient Function include `Example: AWS IAM: Add User To Group`.

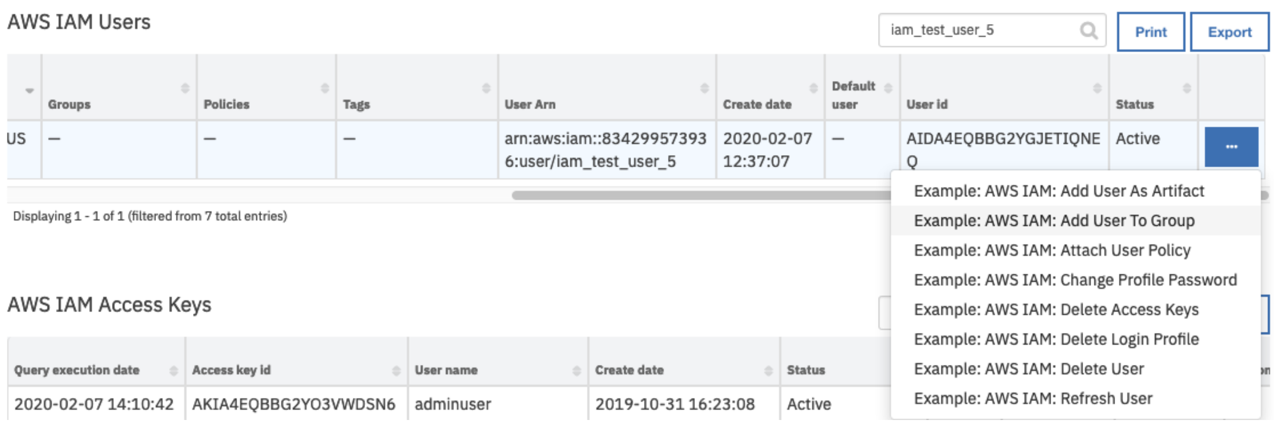
The workflow, `Example: AWS IAM: Add User To Group`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.
- aws_iam_group_names is mapped to activity field which is a drop-down list of group names.

The workflow is initiated by the data table rule, `Example: AWS IAM: Add User To Group`.

1. Open an incident and select the row of data table `AWS IAM Users`corresponding to user who needs to be added to a group.

2. From the selected row's actions menu, select `Example: AWS IAM: Add User To Group`.


AWS IAM Users

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status
US	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_5	2020-02-07 12:37:07	—	AIDA4EQBBG2YGJETIQNEQ	Active

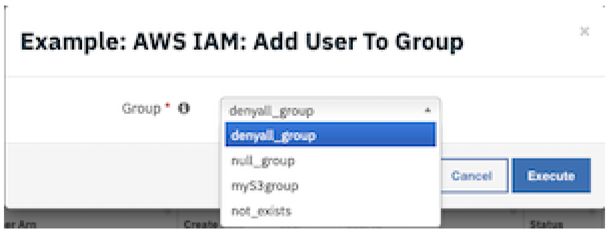
Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

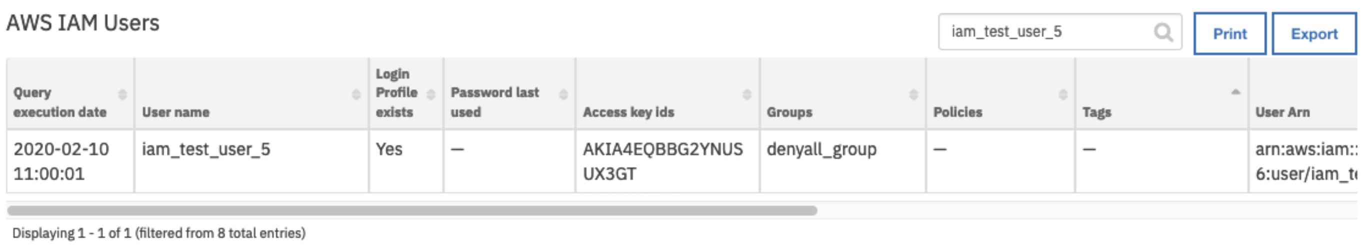
iam_test_user_5
Print
Export
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Refresh User

3. From the drop-down list of user defined groups names , select a group and click Execute.


Example: AWS IAM: Add User To Group

Group * : denyall_group

This invokes the `Example: AWS IAM: Add User To Group` workflow, which calls the `AWS IAM: Add User To Groups` function. On successful completion, for the data table `AWS IAM Users` the `Groups` field will get updated for the selected user.


AWS IAM Users

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:00:01	iam_test_user_5	Yes	—	AKIA4EQBBG2YNUSUX3GT	denyall_group	—	—	arn:aws:iam:: 6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_group_names`	`text`	Yes	`-`	Comma separated list of AWS IAM group names.
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this code block.
    # To see view the output of a Function, run resilient-circuits in DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
inputs.aws_iam_group_names = rule.properties.aws_iam_group
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_add_user_to_groups script ##
# Example result:
"""
Result: {
}
"""
#  Globals
# List of fields in datatable for fn_aws_iam_add_user_to_groups  script
DATA_TBL_FIELDS = ["Groups"]
FN_NAME = "fn_aws_iam_add_user_to_groups"
WF_NAME = "Add User To Group"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    added = 0
    no_such_entity = 0
    added_groups = []
    no_such_entity_groups = []
    if CONTENT:
        for grp_stat in CONTENT:
            if grp_stat["Status"] == "OK":
                added += 1
                added_groups.append(grp_stat["GroupName"])
            else:
                no_such_entity += 1
                no_such_entity_groups.append(grp_stat["GroupName"])
        if added_groups:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
was <b>{1}</b> Groups <b>{2}</b> added " \
                        "for user <b>{3}</b> for Resilient function <b>{4}
</b>."\
                .format(WF_NAME, len(added_groups), ", ".join(str(i) for i
in added_groups), INPUTS["aws_iam_user_name"], FN_NAME)
        if no_such_entity:
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: There
was <b>{1}</b> Groups <b>{2}</b> " \
                        "which did not exist for user <b>{3}</b> for
Resilient function <b>{4}</b>."\
                .format(WF_NAME, len(no_such_entity_groups), ",
".join(str(i) for i in no_such_entity_groups),
INPUTS["aws_iam_user_name"], FN_NAME)
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: List User Groups

Get the IAM groups that the specified IAM user belongs to. Parameter aws_iam_user_name is an IAM user name.

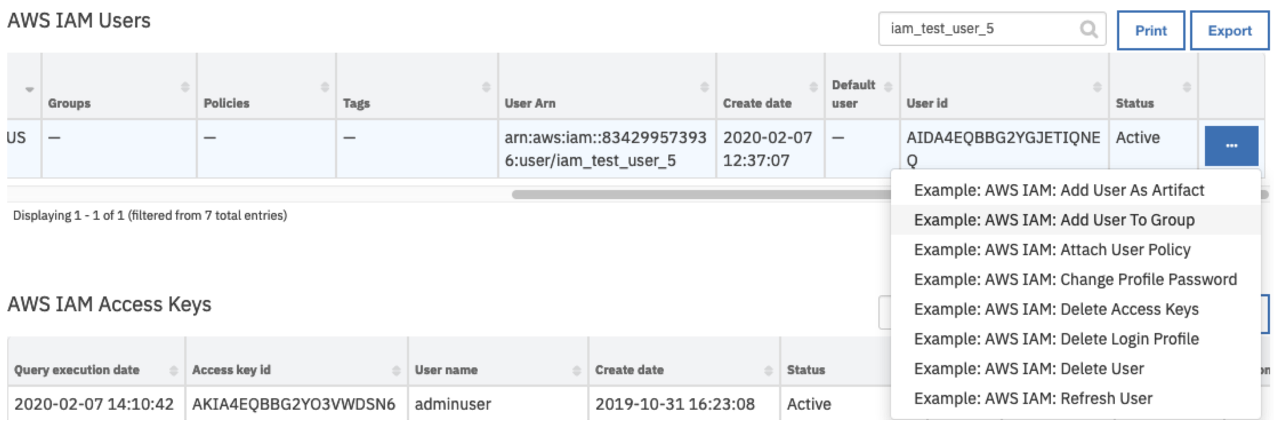
- Example workflows that use this Resilient Function include `Example: AWS IAM: Add User To Group`, `Example: AWS IAM: Refresh User`, `Example: AWS IAM: Remove User From All Groups`, `Example: AWS IAM: Delete User` or , `Example: AWS IAM: Delete User For Artifact`.

The workflow, `Example: AWS IAM: Add User To Group`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.

The workflow is initiated by the data table rule, `Example: AWS IAM: Add User To Group`.

1. Open an incident and select the row of data table `AWS IAM Users`corresponding to user who needs to be added to a group.
2. From the selected row's actions menu, select `Example: AWS IAM: Add User To Group`.


AWS IAM Users

iam_test_user_5 | Print | Export

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
US	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_5	2020-02-07 12:37:07	—	AIDA4EQBBG2YGJETIQNEQ	Active	···

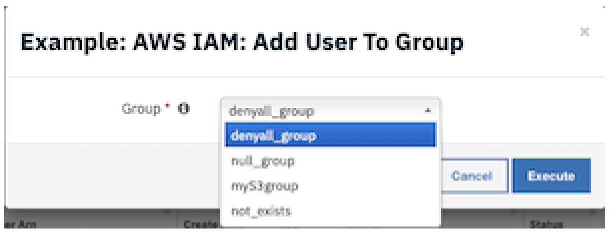
Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

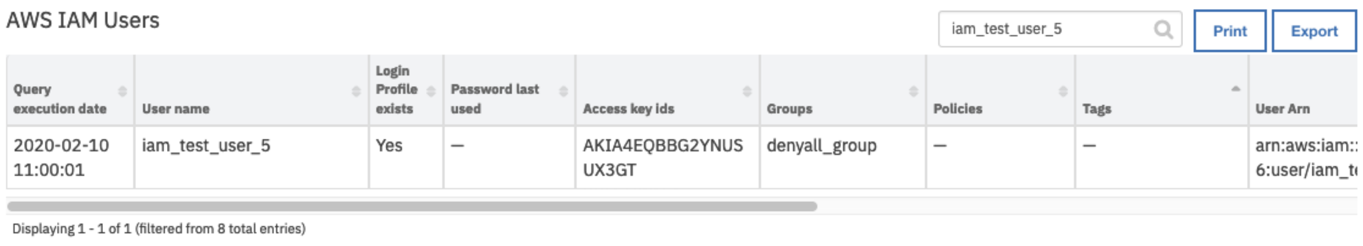
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Refresh User

3. From the drop-down list of user defined groups names , select a group and click Execute.


Example: AWS IAM: Add User To Group

Group * denyall_group

This invokes the `Example: AWS IAM: Remove User From All Groups` workflow, which calls the `AWS IAM: Add User To Groups` function. On successful completion, for the data table `AWS IAM Users` the `Groups` field will get updated for the selected user.


AWS IAM Users

iam_test_user_5 | Print | Export

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:00:01	iam_test_user_5	Yes	—	AKIA4EQBBG2YNUSUX3GT	denyall_group	—	—	arn:aws:iam::6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this code block.
    # To see view the output of a Function, run resilient-circuits in DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_list_user_groups script ##
# Example result:
"""
Result: {
         'version': '1.0', 'success': True, 'reason': None,
         'content': [{'Path': '/', 'GroupName': 'system-admins',
'GroupId': 'AGPAJUCG3BHM64OGVGCBG',
                      'Arn': 'arn:aws:iam::834299573936:group/system-
admins', 'CreateDate': '2017-05-29 20:37:53'}],
                      'raw': '[{"Path": "/", "GroupName": "system-admins",
"GroupId": "AGPAJUCG3BHM64OGVGCBG",
                      "Arn": "arn:aws:iam::834299573936:group/system-
admins", "CreateDate": "2017-05-29 20:37:53"
                      }
                    ]',
         'inputs': {'aws_iam_user_name': 'iam_test_User'},
         'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0', 'host': 'myhost.ibm.com',
                     'execution_time_ms': 1070, 'timestamp': '2019-11-18
10:19:19'
                     }
}
"""
#  Globals
# List of fields in datatable fn_aws_iam_list_user_groups script
DATA_TBL_FIELDS = ["Groups"]
FN_NAME = "fn_aws_iam_list_user_groups"
WF_NAME = "Add User To Group"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>{1}</b> 'Group' result(s) returned for user " \
                    "<b>{2}</b> for Resilient function <b>{3}</b>."\
            .format(WF_NAME, len(CONTENT), INPUTS["aws_iam_user_name"],
FN_NAME)
        groups = []
```

```
            for grp in CONTENT:
                if grp["GroupName"] is not None:
                    groups.append(grp["GroupName"])
            row.Groups = ",".join(groups)
        else:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>no</b> 'Group' result(s) returned for " \
                        "user <b>{1}</b> for Resilient function <b>{2}</b>."\
                .format(WF_NAME, INPUTS["aws_iam_user_name"], FN_NAME)

        incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Delete Login Profile

Delete the password for the specified IAM user, which terminates the user's ability to access AWS services through the AWS Management Console. Parameter aws_iam_user_name is an IAM user name.

- Example workflows that use this Resilient Function include `Example: AWS IAM: Delete Login Profile`, `Example: AWS IAM: Delete User` and `Example: AWS IAM: Delete User For Artifact`.

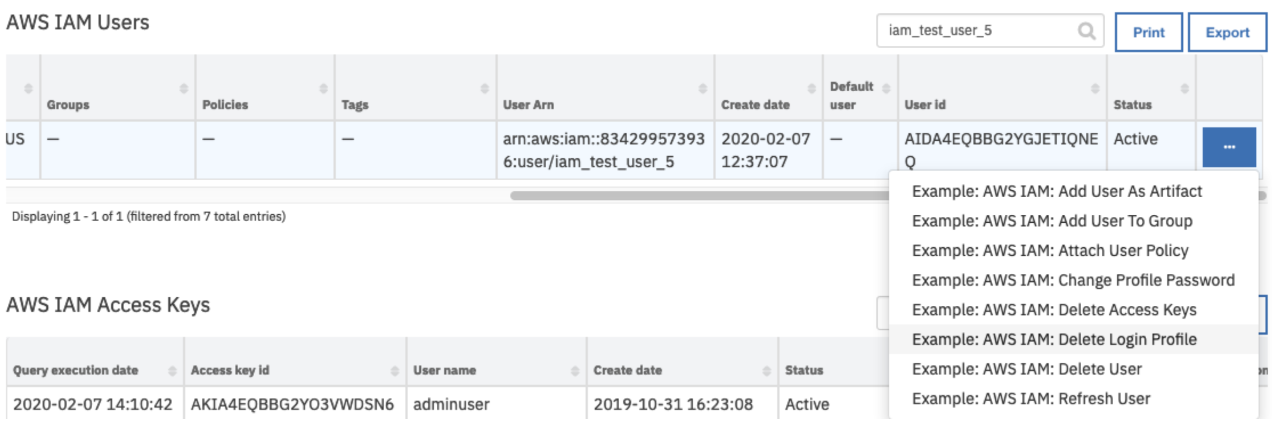
The workflow, `Example: AWS IAM: Delete Login Profile`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.

The workflow is initiated by the data table rule, `Example: AWS IAM: Delete Login Profile`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to the user whose login profile is to be deleted.

2. From the selected row's actions menu, select `Example: AWS IAM: Delete Login Profile`.



AWS IAM Users

iam_test_user_5

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
US	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_5	2020-02-07 12:37:07	—	AIDA4EQBBG2YGJETIQNEQ	Active	···

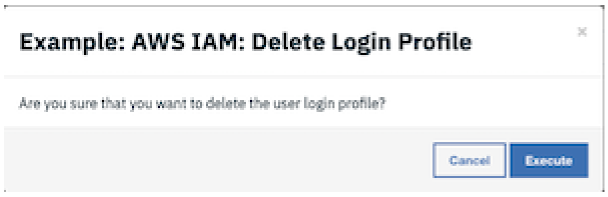
Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

Print
Export
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Refresh User

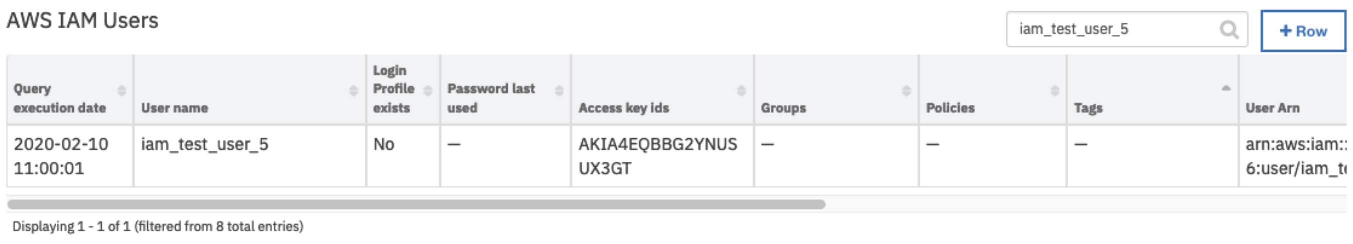
3. User is presented with a warning and an option to Execute or Cancel.



Example: AWS IAM: Delete Login Profile

Are you sure that you want to delete the user login profile?

This invokes the `Example: AWS IAM: Delete Login Profile` workflow, which calls the `AWS IAM: Delete Login Profile` function. On successful completion, for the data table `AWS IAM Users` the `Login Profile exists` field will get updated to "No" for the selected user.



AWS IAM Users

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:00:01	iam_test_user_5	No	—	AKIA4EQBBG2YNUS UX3GT	—	—	—	arn:aws:iam:: 6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this code block.
    # To see view the output of a Function, run resilient-circuits in DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_delete_login_profile script ##
# Example result:
"""
OK
Result: {
          'version': '1.0', 'success': True, 'reason': None,
          'content': 'NoSuchEntity', 'raw': '"NoSuchEntity"',
          'inputs': {'aws_iam_user_name': 'iam_test_User'},
         'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                     'host': 'myhost.ie.ibm.com', 'execution_time_ms':
9170, 'timestamp': '2019-11-18 16:24:17'
                     }
}
NosuchEntity
Result: {
         'version': '1.0', 'success': True, 'reason': None,
         'content': 'OK', 'raw': '"OK"',
         'inputs': {'aws_iam_user_name': 'iam_test_User'},
         'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                     'host': 'myhost.ie.ibm.com', 'execution_time_ms':
9170, 'timestamp': '2019-11-18 16:24:17'
                     }
}
"""
#  Globals
# List of fields in datatable fn_aws_iam_delete_login_profile  script
DATA_TBL_FIELDS = ["Groups"]
FN_NAME = "fn_aws_iam_delete_login_profile"
WF_NAME = "Delete Login Profile"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    if CONTENT:
        if CONTENT == "OK":
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: Login
profile deleted for user <b>{1}</b> for " \
                        "Resilient function <b>{2}</b>.".format(WF_NAME,
INPUTS["aws_iam_user_name"], FN_NAME)
            row.LoginProfileExists = "No"
        elif CONTENT == "NoSuchEntity":
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: Login
profile does not exist for user <b>{1}</b> for " \
                        "Resilient function <b>{2}</b>.".format(WF_NAME,
INPUTS["aws_iam_user_name"], FN_NAME)
            row.LoginProfileExists = "No"
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: List User Policies

Get all managed policies and in-line policies that are attached to the specified IAM user. Parameter aws_iam_user_name is an IAM user name.

- Example workflows that use this Resilient Function include `Example: AWS IAM: Attach User Policy`, `Example: AWS IAM: Refresh User`, `Example: AWS IAM: Get User For Artifact`, `Example: AWS IAM: Detach All User Policies`, `Example: AWS IAM: Delete User` and `Example: AWS IAM: Delete User For Artifact`.

The workflow, `Example: AWS IAM: Attach User Policy`, sets the following input fields for the Function:

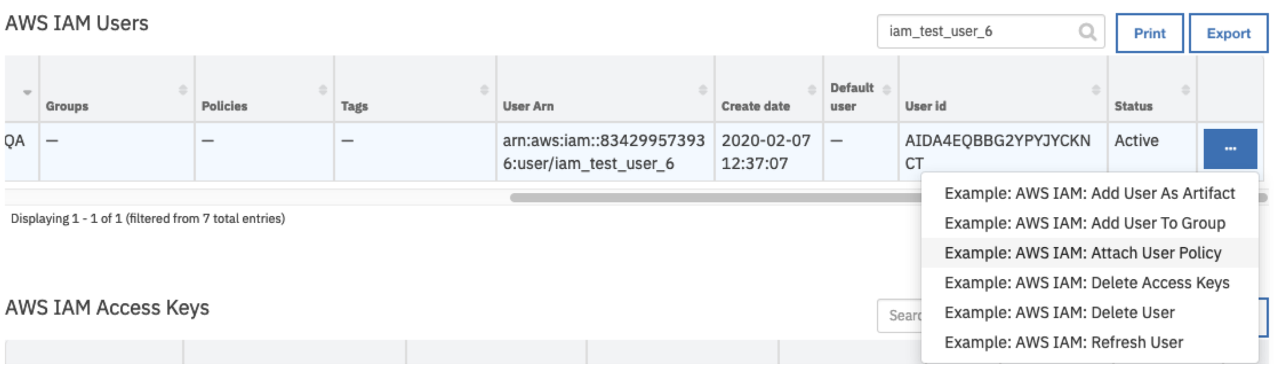
- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.

The workflow, `Example: AWS IAM: Attach User Policy`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.

The workflow is initiated by the data table rule, `Example: AWS IAM: Attach User Policy`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to user who needs to have a policy attached.
2. From the selected rows action menu, select `Example: AWS IAM: Attach User Policy`.



AWS IAM Users

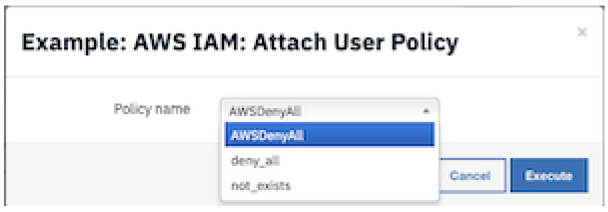
	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
QA	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_6	2020-02-07 12:37:07	—	AIDA4EQBBG2YPYJYCKNCT	Active	···

Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

iam_test_user_6
Print
Export
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete User
Example: AWS IAM: Refresh User

3. From the drop-down list of user defined policty name , select a policy and click Execute.

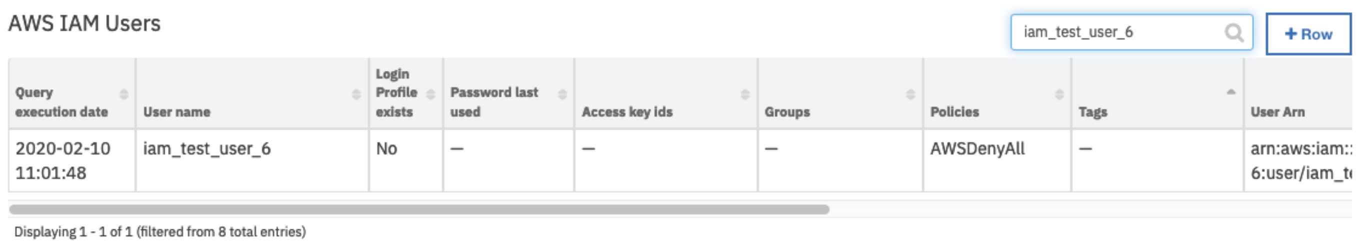


Example: AWS IAM: Attach User Policy

Policy name: AWSDenyAll

This invokes the `Example: AWS IAM: Attach User Policy` workflow, which calls the `AWS IAM: List User Policies` function.

On successful completion, for the data table `AWS IAM Users` the `Policies` field will get updated for the selected user.


AWS IAM Users

iam_test_user_6

+ Row

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-10 11:01:48	iam_test_user_6	No	—	—	—	AWSDenyAll	—	arn:aws:iam:: 6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this code block.
    # To see view the output of a Function, run resilient-circuits in DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_list_user_policies script ##
# Example result:
"""
Result: {
          'version': '1.0', 'success': True, 'reason': None,
          'content': [{'PolicyName': 'test_pol'},
                      {'PolicyName': 'test_pol_2',
                       'PolicyArn': 'arn:aws:iam::834299573936:policy/test_pol_2'},
                      {'PolicyName': 'AmazonRoute53ReadOnlyAccess',
                       'PolicyArn': 'arn:aws:iam::aws:policy/AmazonRoute53ReadOnlyAccess'}],
          'raw': '[{"PolicyName": "test_pol"}, {"PolicyName": "test_pol_2",
                    "PolicyArn": "arn:aws:iam::834299573936:policy/test_pol_2"},
                    {"PolicyName": "AmazonRoute53ReadOnlyAccess",
                    "PolicyArn": "arn:aws:iam::aws:policy/AmazonRoute53ReadOnlyAccess"}]',
          'inputs': {'aws_iam_user_name': 'iam_test_User'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                      'host': 'myhost.ibm.com', 'execution_time_ms':
87423, 'timestamp': '2019-11-21 11:55:29'
                     }
}
"""
#  Globals
# List of fields in datatable for fn_aws_iam_list_user_policies script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_list_user_policies"
WF_NAME = "Example: AWS IAM: Attach User Policy"
# Processing
CONTENT = results.content
INPUTS = results.inputs
note_text = ''

def main():
    note_text = ''
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>{1}</b> 'Policy name' result(s) returned for user " \
                    "<b>{2}</b> for Resilient function <b>{3}</b>."\
            .format(WF_NAME, len(CONTENT), INPUTS["aws_iam_user_name"],
FN_NAME)
        policy_names = []
        for pol in CONTENT:
            if pol["PolicyName"] is not None:
                policy_names.append(pol["PolicyName"])
        row.Policies = ",".join(policy_names)
    else:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>no</b> 'Policy name' result(s) returned for " \
                    "user <b>{1}</b> for Resilient function <b>{2}</b>."\
            .format(WF_NAME, INPUTS["aws_iam_user_name"], FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: Detach User policies

Remove the specified managed policy from the specified IAM user. Parameter aws_iam_user_name is an IAM user name. Parameter aws_iam_policy_names (optional) is a comma separated list of IAM policy names. Parameter (optional) aws_iam_arns is a comma separated list of IAM policy arns.

Note: A user can also have inline policies embedded with it, this function will delete inline policies associated with the the user. Note: one of parameters aws_iam_policy_names or aws_iam_arns required to be set.

- Example workflows that use this Resilient Function include `Example: AWS IAM: Detach All User Policies`, `Example: AWS IAM: Delete User For Artifact` or `Example: AWS IAM: Delete User`.

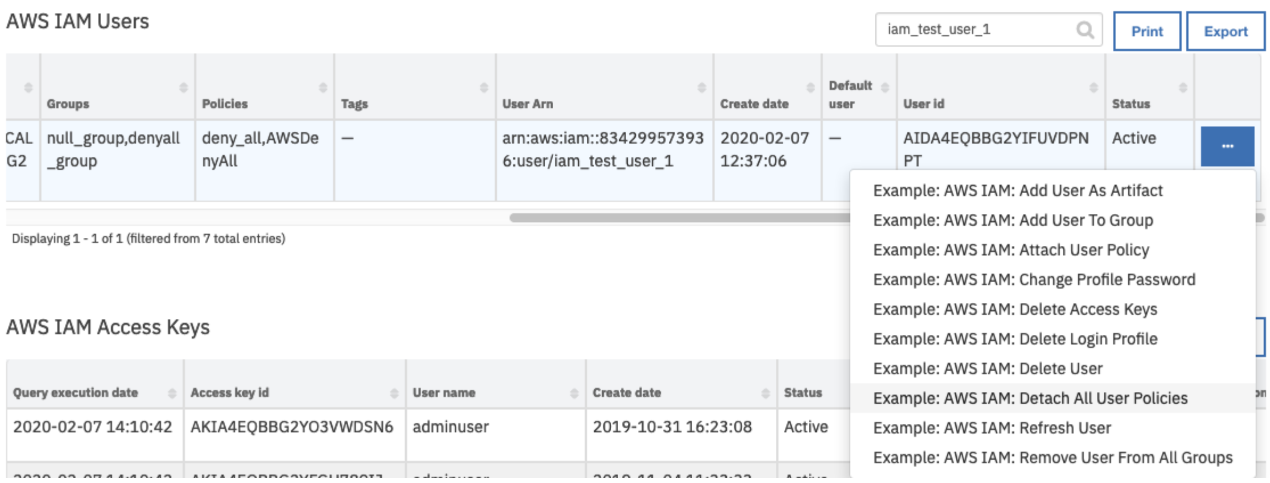
The workflow, `Example: AWS IAM: Detach All User Policies`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected row of data table `AWS IAM Users`.
- aws_iam_policy_names is mapped to all policy names for the user from the selected row of data table `AWS IAM Users`.

The workflow is initiated by the data table rule, `Example: AWS IAM: Detach All User Policies`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to the user who needs to have all polices removed or deleted.

2. From the selected row's actions menu, select `Example: AWS IAM: Detach All User Policies`.



AWS IAM Users

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status
CAL G2	null_group,denyall _group	deny_all,AWSDe nyAll	—	arn:aws:iam::83429957393 6:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPN PT	Active

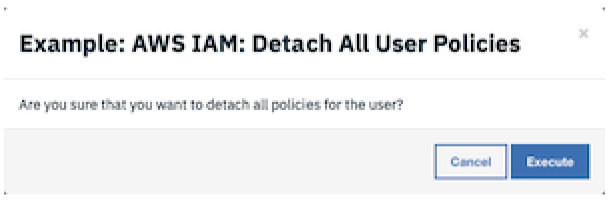
Displaying 1 - 1 of 1 (filtered from 7 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

iam_test_user_1
Print
Export
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Detach All User Policies
Example: AWS IAM: Refresh User
Example: AWS IAM: Remove User From All Groups

3. User is presented with a warning and an option to Execute or Cancel.

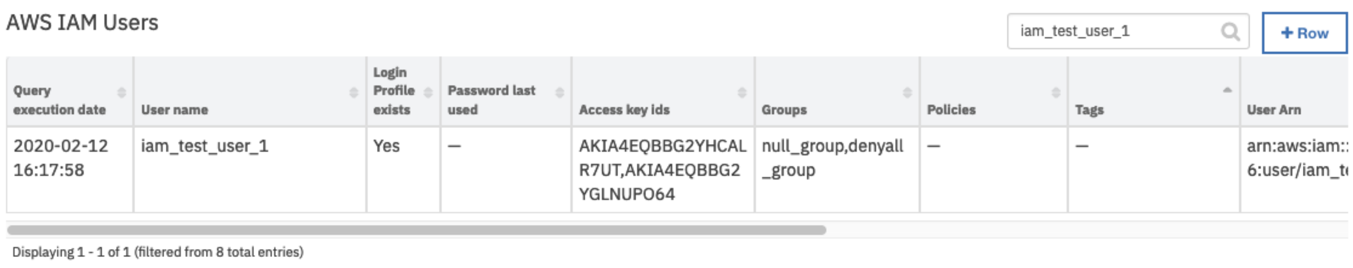


Example: AWS IAM: Detach All User Policies

Are you sure that you want to detach all policies for the user?

This invokes the `Example: AWS IAM: Detach All User Policies` workflow, which calls the `AWS IAM: Detach User policies` function.

On successful completion, for the data table `AWS IAM Users` the `Policies` field will get updated to an empty value for the selected user.



AWS IAM Users

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-12 16:17:58	iam_test_user_1	Yes	—	AKIA4EQBBG2YHCAL R7UT,AKIA4EQBBG2 YGLNUPO64	null_group,denyall _group	—	—	arn:aws:iam:: 6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_arns`	`text`	No	`-`	Comma separated list of AWS IAM Amazon Resource Names (ARNs).
`aws_iam_policy_names`	`text`	No	`-`	Comma separated list of AWS IAM policy names.
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

Note: At least One of the parameters `aws_iam_arns` or `aws_iam_user_name` must be set.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this
code block.
    # To see view the output of a Function, run resilient-circuits in
DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
inputs.aws_iam_policy_names = row.Policies
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_detach_user_policies script ##
# Example result:
"""
OK
Result: {
          'version': '1.0', 'success': True, 'reason': None,
          'content': [{'PolicyName': 'AWSDenyAll', 'Status': 'OK'}
                      {'PolicyName': 'AWSDenyAll_2', 'Status':
'NoSuchEntity'}],
          'raw': '[{'PolicyName': "AWSDenyAll", 'Status": 'OK'},
                  {'PolicyName': 'AWSDenyAll_2', 'Status':
'NoSuchEntity'}]',
          'inputs': {'aws_iam_arns': 'arn:aws:iam::aws:policy/AWSDenyAll',
'aws_iam_user_name': 'iam_test_User_1'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0', 'host': 'myhost.ibm.com',
                      'execution_time_ms': 790, 'timestamp': '2019-11-29
12:18:30'
                     }
}
"""
#  Globals
# List of fields in datatable for fn_aws_iam_detach_user_policies  script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_detach_user_policies"
WF_NAME = "Detach All User Policies"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():
    note_text = ''
    added = 0
    no_such_entity = 0
    added_policies = []
    no_such_entity_policies = []
    if CONTENT:
        for pol_stat in CONTENT:
            if pol_stat["Status"] == "OK":
                added += 1
                added_policies.append(pol_stat["PolicyName"])
            else:
                no_such_entity += 1
                no_such_entity_policies.append(pol_stat["PolicyName"])
        if added_policies:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> Policies <b>{2}</b> detached " \
                        "for user <b>{3}</b> for Resilient function <b>{4}
</b>."\
                .format(WF_NAME, len(added_policies), ", ".join(str(i) for
i in added_policies), INPUTS["aws_iam_user_name"], FN_NAME)
        if no_such_entity:
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: There
were <b>{1}</b> Policies <b>{2}</b> " \
                        "which did not exist for user <b>{3}</b> for
Resilient function <b>{4}</b>."\
                .format(WF_NAME, len(no_such_entity_policies), ",
".join(str(i) for i in no_such_entity_policies),
INPUTS["aws_iam_user_name"], FN_NAME)
    else:
        note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There was
no result returned for Resilient function <b>{0}</b>."\
            .format(WF_NAME, FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Function - AWS IAM: List User Access Key Ids

Get information about the access key IDs associated with the specified IAM user. Parameter aws_iam_user_name is an IAM user name.

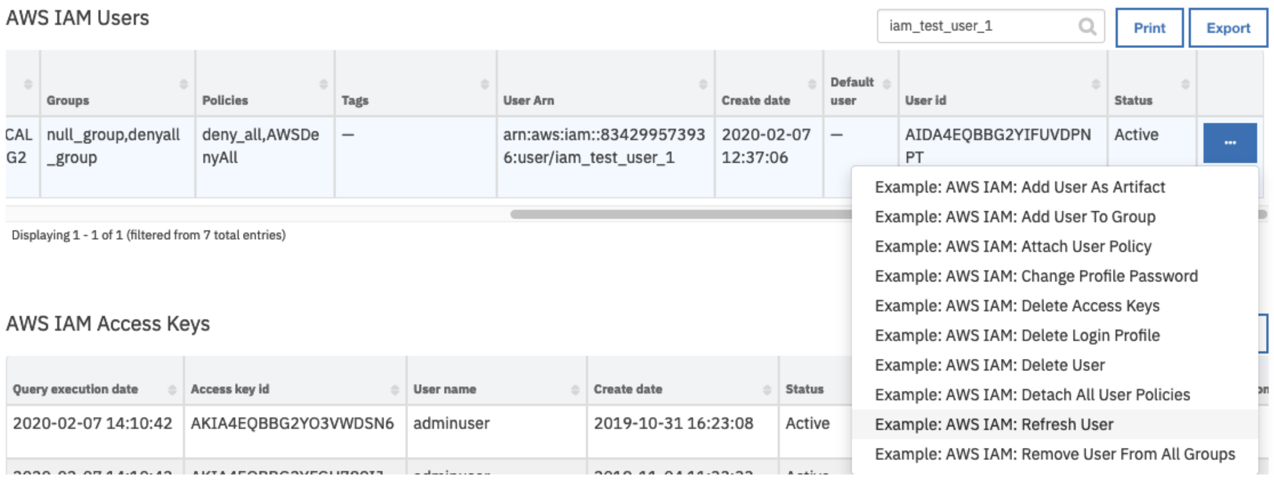
- Example workflows that use this Resilient Function include `Example: AWS IAM: Delete Access Keys`, `Example: AWS IAM: Refresh User`, `Example: AWS IAM: Get User For Artifact`, `Example: AWS IAM: Delete User` or `Example: AWS IAM: Delete User For Artifact`.

The workflow, `Example: AWS IAM: Refresh User`, sets the following input fields for the Function:

- aws_iam_user_name is mapped to a user name from the selected data table row.

The workflow is initiated by the data table rule, `Example: AWS IAM: Refresh User`.

1. Open an incident and select the row of data table `AWS IAM Users` corresponding to the user which is to have its properties refreshed for th dat table.
2. From the selected row's actions menu, select `Example: AWS IAM: Refresh User`. fn-aws-iam-list-user-access-key-ids


AWS IAM Users

iam_test_user_1 | Print | Export

	Groups	Policies	Tags	User Arn	Create date	Default user	User id	Status	
CAL G2	null_group,denyall _group	deny_all,AWSDe nyAll	—	arn:aws:iam::83429957393 6:user/iam_test_user_1	2020-02-07 12:37:06	—	AIDA4EQBBG2YIFUVDPN PT	Active	···

Displaying 1 - 1 of 1 (filtered from 7 total entries)

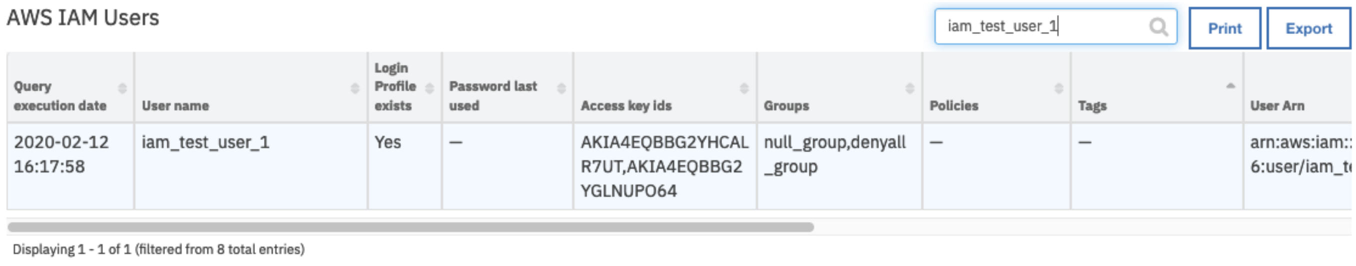
Example: AWS IAM: Add User As Artifact
Example: AWS IAM: Add User To Group
Example: AWS IAM: Attach User Policy
Example: AWS IAM: Change Profile Password
Example: AWS IAM: Delete Access Keys
Example: AWS IAM: Delete Login Profile
Example: AWS IAM: Delete User
Example: AWS IAM: Detach All User Policies
Example: AWS IAM: Refresh User
Example: AWS IAM: Remove User From All Groups

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status
2020-02-07 14:10:42	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active

Pressing Execute invokes the `Example: AWS IAM: Refresh User` workflow, which calls the `AWS IAM: List User Access Key Ids` function.

On successful completion, for the data table `AWS IAM Users` the `Access key ids` field will get updated for the selected user.


AWS IAM Users

iam_test_user_1 | Print | Export

Query execution date	User name	Login Profile exists	Password last used	Access key ids	Groups	Policies	Tags	User Arn
2020-02-12 16:17:58	iam_test_user_1	Yes	—	AKIA4EQBBG2YHCAL R7UT,AKIA4EQBBG2 YGLNUPO64	null_group,denyall _group	—	—	arn:aws:iam:: 6:user/iam_t

Displaying 1 - 1 of 1 (filtered from 8 total entries)

▶ Inputs:

Name	Type	Required	Example	Tooltip
`aws_iam_user_name`	`text`	Yes	`AWS IAM user name`	AWS IAM user name.

▶ Outputs:

```
results = {
    # TODO: Copy and paste an example of the Function Output within this
code block.
    # To see view the output of a Function, run resilient-circuits in
DEBUG mode and invoke the Function.
    # The Function results will be printed in the logs: "resilient-
circuits run --loglevel=DEBUG"
}
```

▶ Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
```

▶ Example Post-Process Script:

```
##  AWS IAM - fn_aws_iam_list_user_access_keys script ##
# Example result:
"""
Result: {
          'version': '1.0', 'success': True, 'reason': None,
          'content': [{'UserName': 'iam_test_User', 'AccessKeyId':
'AKIA4EQBBG2YKXYJB55L',
                       'Status': 'Active', 'CreateDate': '2019-11-12
11:09:38'
                       }],
          'raw': '[{"UserName": "iam_test_User", "AccessKeyId":
"AKIA4EQBBG2YKXYJB55L",
                  "Status": "Active", "CreateDate": "2019-11-12
11:09:38"}]',
          'inputs': {'aws_iam_user_name': 'iam_test_User'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                      'host': 'myhost.ibm.com', 'execution_time_ms': 5365,
'timestamp': '2019-11-21 10:41:22'}}
"""
#  Globals
# List of fields in datatable fn_aws_iam_list_user_access_keys script
DATA_TBL_FIELDS = ["AccessKeyIds"]
FN_NAME = "fn_aws_iam_list_user_access_keys"
WF_NAME = "Refresh User"
# Processing
CONTENT = results.content
```

```
INPUTS = results.inputs
note_text = ''

def main():
    note_text = ''
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>{1}</b> 'Access key' result(s) returned for user " \
                    "<b>{2}</b> for Resilient function <b>{3}</b>."\
            .format(WF_NAME, len(CONTENT), INPUTS["aws_iam_user_name"],
FN_NAME)
        access_key_ids = []
        for ak_id in CONTENT:
            if ak_id["AccessKeyId"] is not None:
                access_key_ids.append(ak_id["AccessKeyId"])
        row.AccessKeyIds = ",".join(access_key_ids)
    else:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>no</b> 'Access key' result(s) returned for " \
                    "user <b>{1}</b> for Resilient function <b>{2}</b>."\
            .format(WF_NAME, INPUTS["aws_iam_user_name"], FN_NAME)

    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()
```

Data Table - AWS IAM Access Keys

screenshot: dt-aws-iam-access-keys

API Name:

aws_iam_access_keys

Columns:

Column Name	API Access Name	Type	Tooltip
Access key id	`AccessKeyId`	`text`	-
Create date	`CreateDate`	`text`	-
Default key	`DefaultKey`	`text`	-
Last used date	`LastUsedDate`	`text`	-
Region	`Region`	`text`	-
Service name	`ServiceName`	`text`	-
Status	`Status`	`text`	-

Column Name	API Access Name	Type	Tooltip
User name	`UserName`	`text`	-
Query execution date	`query_execution_date`	`text`	-

Data Table - AWS IAM Users

screenshot: dt-aws-iam-users

API Name:

aws_iam_users

Columns:

Column Name	API Access Name	Type	Tooltip
Access key ids	`AccessKeyIds`	`text`	-
User Arn	`Arn`	`text`	-
Create date	`CreateDate`	`text`	-
Default user	`DefaultUser`	`text`	-
Groups	`Groups`	`text`	-
Login Profile exists	`LoginProfileExists`	`text`	-
Password last used	`PasswordLastUsed`	`text`	-
Policies	`Policies`	`text`	-
Status	`Status`	`text`	-
Tags	`Tags`	`text`	-
User id	`UserId`	`text`	-
User name	`UserName`	`text`	-
Query execution date	`query_execution_date`	`text`	-

Custom Artifact Types

Display Name	API Access Name	Description
AWS IAM User Name	`aws_iam_user_name`	Amazon Web Services (AWS) IAM user name.
AWS IAM Access Key ID	`aws_iam_access_key_id`	Amazon Web Services (AWS) IAM access key id.

Rules

Rule Name	Object	Workflow Triggered
Example: AWS IAM: List Access Keys	incident	`wf_aws_iam_list_access_keys`
Example: AWS IAM: Add Access Key As Artifact	aws_iam_access_keys	`-`
Example: AWS IAM: Delete Access Key For Artifact	artifact	`wf_aws_iam_delete_access_key_for_artifact`
Example: AWS IAM: Remove User From All Groups	aws_iam_users	`wf_aws_iam_remove_user_from_all_groups`
Example: AWS IAM: Delete Access Keys	aws_iam_users	`wf_aws_iam_delete_access_keys`
Example: AWS IAM: Add User To Group	aws_iam_users	`wf_aws_iam_add_user_to_group`
Example: AWS IAM: Delete Access Key	aws_iam_access_keys	`wf_aws_iam_delete_access_key`
Example: AWS IAM: Add User As Artifact	aws_iam_users	`-`
Example: AWS IAM: Delete Login Profile	aws_iam_users	`wf_aws_iam_delete_login_profile`
Example: AWS IAM: Delete User For Artifact	artifact	`wf_aws_iam_delete_user_for_artifact`
Example: AWS IAM: Get User For Artifact	artifact	`wf_aws_iam_get_user_for_artifact`
Example: AWS IAM: Delete User	aws_iam_users	`wf_aws_iam_delete_user`
Example: AWS IAM: Attach User Policy	aws_iam_users	`wf_aws_iam_attach_user_policy`
Example: AWS IAM: List Users	incident	`wf_aws_iam_list_users`
Example: AWS IAM: Refresh User	aws_iam_users	`wf_aws_iam_refresh_user`
Example: AWS IAM: Get access Key For Artifact	artifact	`wf_aws_iam_get_access_key_for_artifact`

Rule Name	Object	Workflow Triggered
Example: AWS IAM: Change Profile Password	aws_iam_users	`wf_aws_iam_change_profile_password`
Example: AWS IAM: Detach All User Policies	aws_iam_users	`wf_aws_iam_detach_all_user_policies`