

User Guide: fn_aws_iam_v1.0.1

Table of Contents

- [Key Features](#)
- [Function - AWS IAM: List Users](#)
- [Function - AWS IAM: Delete User](#)
- [Function - AWS IAM: List User Access Key Ids](#)
- [Function - AWS IAM: Update Access Key](#)
- [Function - AWS IAM: Delete Access Keys](#)
- [Function - AWS IAM: List User Groups](#)
- [Function - AWS IAM: Add User To Groups](#)
- [Function - AWS IAM: Remove User From Groups](#)
- [Function - AWS IAM: List User Policies](#)
- [Function - AWS IAM: Attach User policies](#)
- [Function - AWS IAM: Detach User policies](#)
- [Function - AWS IAM: List SSH Public Keys](#)
- [Function - AWS IAM: Delete SSH Public Keys](#)
- [Function - AWS IAM: List Service Specific Credentials](#)
- [Function - AWS IAM: Delete Service Specific Credentials](#)
- [Function - AWS IAM: List Signing Certificates](#)
- [Function - AWS IAM: Delete Signing Certificates](#)
- [Function - AWS IAM: List MFA Devices](#)
- [Function - AWS IAM: Deactivate MFA Devices](#)
- [Function - AWS IAM: Delete Virtual MFA Devices](#)
- [Function - AWS IAM: Update Login Profile](#)
- [Function - AWS IAM: Delete Login Profile](#)
- [Data Table - AWS IAM Access Keys](#)
- [Data Table - AWS IAM Users](#)
- [Custom Artifact Types](#)
- [Rules](#)

Key Features

Amazon Web Services Identity and Access Management (AWS IAM) allows management of access to AWS services and resources securely. You can use IAM to create and manage AWS users and groups, and use permissions to allow or deny access to AWS resources. The AWS IAM integration with the Resilient platform allows you to query and update users or access keys for an AWS account.

You can execute the following types of queries:

- Get a list of users and associated items (login profile, access keys, groups, policies).
- Get a list of access keys.
- List objects associated with a user:
 - Access keys.

- Groups.
- Policies.
- SSH public keys.
- Service-specific credentials.
- Signing certificates.
- Active MFA devices (Virtual devices flagged).

You can also use the integration to make the following changes to an AWS IAM environment:

- Delete a user and delete or remove items associated with the user.
 - Attach a user policy.
 - Detach all policies for a user.
 - Add a 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.
 - Delete all SSH Public Keys for a user.
 - Delete all service-specific credentials for a user.
 - Delete all signing certificates for a user.
 - De-activate all active MFA devices for a user.
 - Delete all active MFA virtual devices for a user.
-

Function - AWS IAM: List Users

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

AWS IAM Demo

Description

No description.

Tasks Details Breach Notes Members News Feed Attachments Stats Timeline
Artifacts Email

Actions ▾

Example: AWS IAM: List Access Keys
Example: AWS IAM: List Users

Action Status
Workflow Status
Close Incident
Delete Incident
Data Generated

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.

Example: AWS IAM: List Users

User filter ⓘ

Group filter ⓘ

Policy filter ⓘ

Access Key filter ⓘ

3. Press Execute to invoke the `Example: AWS IAM: List Users` workflow, which calls the `AWS IAM: List Users` function.

On successful completion of the workflow, the data table `AWS IAM Users` is updated in the Resilient platform with the users' properties for the selected AWS account.

AWS IAM Users

Search...

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status	
2020-03-20 15:15:06	adminuser	Yes	AKIA4EQBBG2YO3V WDSN6,AKIA4EQBBG2YP4S4SOEV	system-admins	test_pol,test_pol_2,AmazonRoute53ReadOnlyAccess	Name,Email,Environment,Account_Type	arn:aws:iam::834299573936:user/adminuser	2019-10-31 16:23:07	Yes	Active	⋮
2020-03-20 15:15:06	iam_test_user_1	Yes	AKIA4EQBBG2YGLN UPO64	null_group,denyall_group,myS3group	deny_all,AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_1	2020-02-07 12:37:06	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4 MROM	—	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_11	Yes	AKIA4EQBBG2YDGC EJTL	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_11	2020-03-11 16:49:38	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_12	Yes	AKIA4EQBBG2YM2XI TV44	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_12	2020-03-11 16:49:38	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_2	Yes	AKIA4EQBBG2YCINN CIYP5,AKIA4EQBBG2YIFZWDMY	null_group,lggroup, denyall_group	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_2	2020-02-07 12:37:06	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_3	Yes	AKIA4EQBBG2YFDH XQC42,AKIA4EQBBG2YFBVKMMSH	myS3group	inline_policy,AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_3	2020-02-07 12:37:06	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_4	Yes	AKIA4EQBBG2YIAC VUF,AKIA4EQBBG2YMOQWX6F	myS3group	deny_all	—	arn:aws:iam::834299573936:user/iam_test_user_4	2020-02-07 12:37:06	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_5	Yes	AKIA4EQBBG2YL57R 2BX3,AKIA4EQBBG2YKN74TQHS	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_5	2020-03-11 16:50:35	—	Active	⋮
2020-03-20 15:15:06	iam_test_user_7	Yes	AKIA4EQBBG2YNS3 4245,AKIA4EQBBG2Y06VND6P3	null_group,denyall_group	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_7	2020-02-19 11:04:33	—	Active	⋮

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

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_access_key_filter</code>	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.
<code>aws_iam_group_filter</code>	text	No	–	Filter users based on group membership. Filter by group name. Can be a string or regular expression.
<code>aws_iam_policy_filter</code>	text	No	–	Filter users based on policies applied to user. Filter by policy name. Can be a string or regular expression.
<code>aws_iam_query_type</code>	select	No	–	Type of query to perform for list_users, can be one of 'users' or 'access_keys'. Optional parameter.
<code>aws_iam_user_filter</code>	text	No	–	Filter users or access keys based on user name. Can be a string or regular expression.
<code>aws_iam_user_name</code>	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': 'AIDA4EQBBG2YGZOXT2JB',
'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":
"AIDA4EQBBG2YGGZ0QXT2JB", "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", "Arn",
"DefaultUser", "CreateDate", "LoginProfileExists",
"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 = u''
    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> user(s) returned for Resilient function " \
                    "<b>{2}</b>.".format(WF_NAME, len(CONTENT), FN_NAME)
        note_text += "<br>Adding new row(s) to data table <b>{0}</b> for
<b>{1}</b> user(s).<br>".format("AWS IAM Users", len(CONTENT))
        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 += u"<br>aws_iam_user_filter: <b>{0}</b>"

```

```

</br>".format(INPUTS["aws_iam_user_filter"])
    if INPUTS.get("aws_iam_group_filter"):
        note_text += u"<br>aws_iam_group_filter: <b>{0}</b>"
</br>".format(INPUTS["aws_iam_group_filter"])
    if INPUTS.get("aws_iam_policy_filter"):
        note_text += u"<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 += u"<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()

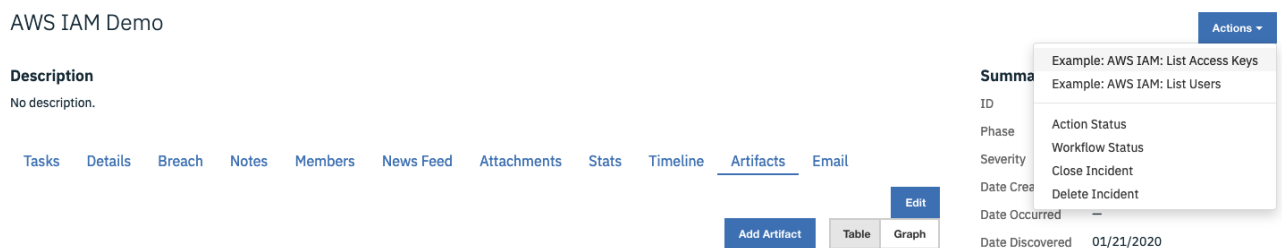
```

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.

3. Press Execute to invoke the **Example: AWS IAM: List Access keys** workflow, which calls the **AWS IAM: List Users** function.

On successful completion of the workflow, the data table **AWS IAM Access Keys** is updated in the Resilient platform with the users' properties for the selected AWS account.

AWS IAM Access Keys

  [+ 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	AKIA4EQBBG2Y03VWDSN6	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	AKIA4EQBBG2YHICALR7UT	iam_test_user_1	2020-02-07 12:37:08	Active	—	—	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YCNCIYP5	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	AKIA4EQBBG2YHQAOWDW O	iam_test_user_6	2020-02-07 12:37:09	Active	—	—	N/A
2020-02-07 14:10:42	AKIA4EQBBG2YGLNUP064	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 is indicated by "Yes" in the "Default key" field.

► Inputs:

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

Name	Type	Required	Example	Tooltip
<code>aws_iam_user_name</code>	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': 'AIDA4EQBBG2YD0LTU6QSM',
                'Arn':
'arn:aws:iam::123456789123:user/iam_test_User', 'CreateDate': '2019-11-05
15:54:43'},
                { 'Path': '/', 'UserName': 'iam_test_User_2',
'UserId': 'AIDA4EQBBG2YGZ0QXT2JB',
                '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":
"AIDA4EQBBG2YD0LTU6QSM", "Arn":
"arn:aws:iam::834299573936:user/iam_test_User", "CreateDate": "2019-11-05
15:54:43"}, {"Path": "/", "UserName": "iam_test_User_2", "UserId":
"AIDA4EQBBG2YGZ0QXT2JB", "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 = u''
    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:
        key_count = 0
        for u in CONTENT:
            if u["AccessKeyIds"]:
                for k in u["AccessKeyIds"]:
                    key_count += 1
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>{1}</b> access keys(s) returned for Resilient function " \
                    "<b>{2}</b>.".format(WF_NAME, key_count, FN_NAME)
        note_text += "<br>Adding new rows to data table <b>{0}</b> for <b>
{1}</b> access keys(s).<br>".format("AWS IAM Access Keys", key_count)
        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 += u"<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 += u"<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 deletes or removes the following items attached to the user:

```

Password
Access keys
Inline policies
Attached managed policies
Group memberships
Signing certificates
SSH public keys
Service specific credentials
Deactivate Multi-factor authentication (MFA) devices
Delete virtual MFA Devices

```

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 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 field 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_13 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status	
2020-03-11 16:49:38	iam_test_user_13	Yes	AKIA4EQBBG2YOS74BMU6,AKIA4EQBBG2YAXR6ISXO	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_13	2020-03-11 16:49:38	—	Active	...

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name
2020-03-11 17:01:04	AKIA4EQBBG2YO3VWDSN6	jprendergast	2019-10-31 16:23:08	Active	—	—	N/A

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: Get Access Keys
 Example: AWS IAM: Refresh User

The 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?
 Note: The following items in AWS IAM if associated with the user will be removed, de-activated or deleted:

1. Delete login profile.
2. Delete access keys.
3. Delete inline policies.
4. Detach managed policies.
5. Remove user from groups.
6. Delete signing certificates.
7. Delete SSH public keys.
8. Delete service specific credentials.
9. De-activate MFA devices.
10. Delete virtual MFA devices.

Cancel Execute

3. Press Execute to invoke the **Example: AWS IAM: Delete User** workflow, which calls the **AWS IAM: Delete User** function.

On successful completion of the workflow, the data table **AWS IAM Users** is refreshed in the Resilient platform with the updated access key details for the selected user. The **Status** field of the data table transitions to **Deleted**.

AWS IAM Users

iam_test_user_13 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 11:18:26	iam_test_user_13	No	—	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_13	2020-03-11 16:49:38	—	Deleted

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**.

Show 25

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	

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

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

Example: AWS IAM: Delete User For Artifact

Are you sure that you want to delete the user?
 Note: The following items in AWS IAM if associated with the user will be removed, de-activated or deleted:

1. Delete login profile.
2. Delete access keys.
3. Delete inline policies.
4. Detach managed policies.
5. Remove user from groups.
6. Delete signing certificates.
7. Delete SSH public keys.
8. Delete service specific credentials.
9. De-activate MFA devices.
10. Delete virtual MFA devices.

Cancel
 Execute

3. Press Execute to invoke the **Example: AWS IAM: Delete User For Artifact** workflow, which calls the **AWS IAM: Delete User** function.

On successful completion of the workflow, the artifact description is updated with details of the user deletion.

iam_test_user_6

Details
 Edit

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
------	------	----------	---------	---------

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

► Outputs:

```
results = { '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'
            }
}
```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```
## AWS IAM - fn_aws_iam_delete_access_keys script ##
# 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: List User Access Key IDs

Gets 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 field 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 the data table.
2. From the selected row's actions menu, select [Example: AWS IAM: Refresh User](#).

AWS IAM Users

iam_test_user_10 Print Export

date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 17:01:04	iam_test_user_10	Yes	AKIA4EQBBG2YGRF WXAP	denyall_group,mys3group	deny_all,AWSDe nyAll	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date
2020-03-11 17:01:04	AKIA4EQBBG2Y03VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—
2020-03-11 17:01:04	AKIA4EQBBG2YP4S4SOEV	adminuser	2019-11-04 11:33:33	Active	Yes	2020-03-11 16:52:00

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: Get Access Keys
Example: AWS IAM: Refresh User
Example: AWS IAM: Remove User From All Groups

3. Press Execute to invoke the [Example: AWS IAM: Refresh User](#) workflow, which calls the [AWS IAM: List User Access Key IDs](#) function.

On successful completion of the workflow, the `Access key ids` field of the [AWS IAM Users](#) data table is updated for the selected user.

AWS IAM Users

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 14:50:05	iam_test_user_10	Yes	AKIA4EQBBG2YGRWF WXAP,AKIA4EQBBG2 YC3T4MROM	denyall_group,mys 3group	deny_all,AWSDe nyAll	—	arn:aws:iam::83429957393 6:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

► Inputs:

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

► Outputs:

```

results = {
    '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' }
}

```

► 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 ##
# 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():

    if CONTENT:
        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:
            row.AccessKeyIds = ""

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

Function - AWS IAM: Update Access Key

Change the status of an access key from Active to Inactive, or vice versa. Parameter `aws_iam_user_name` is an IAM user name. Parameter `aws_iam_access_key_id` is an IAM user access key id. Parameter `aws_iam_status` is be set to "Active" or "Inactive" to change the status of the access key.

Example workflows that use this Resilient function include [Example: AWS IAM: Deactivate Access Key](#).

The workflow, [Example: AWS IAM: Deactivate Access Key](#), sets the following input field for the function:

- `aws_iam_user_name` is mapped to a user name from the selected data table row.
- `aws_iam_access_key_id` is mapped to an access key ID from the selected data table row.
- `aws_iam_status` is set to the value "Inactive".

The workflow is initiated by the data table rule, [Example: AWS IAM: Deactivate Access Key](#).

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 the data table.
2. From the selected row's actions menu, select [Example: AWS IAM: Deactivate Access Key](#).

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name	Region	
2020-03-09 10:52:21	AKIA4EQBBG2YGRFWXAP	iam_test_user_10	2020-03-05 17:17:02	Active	—	—	N/A	N/A	...

Displaying 1 - 1 of 1 (filtered from 15 total entries)

Example: AWS IAM: Add Access Key As Artifact
 Example: AWS IAM: Deactivate Access Key
 Example: AWS IAM: Delete Access Key
 Example: AWS IAM: Get User
 Example: AWS IAM: Refresh Access Key

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

Example: AWS IAM: Deactivate Access Key
✕

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

Cancel
Execute

- Press Execute to invoke the **Example: AWS IAM: Deactivate Access Key** workflow, which calls the **AWS IAM: Update Access Key** function.

On successful completion of the workflow, the **Status** field of the **AWS IAM Access Keys** data table is transitioned to **Inactive**

AWS IAM Access Keys

AWS IAM Access Keys

AKIA4EQBBG2YGRFWXAP

Print

Export

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name	Region	
2020-03-09 10:52:21	AKIA4EQBBG2YGRFWXAP	iam_test_user_10	2020-03-05 17:17:02	Inactive	—	—	N/A	N/A	...

► Inputs:

Name	Type	Required	Example	Tooltip
aws_iam_access_key_id	text	No	—	—
aws_iam_status	select	No	—	—
aws_iam_user_name	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```
results = {'inputs': {'aws_iam_user_name': 'iam_test_user',
'aws_iam_access_key_id': 'AKIA4EQBBG2YGLNUP064',
'aws_iam_status': 'Inactive'},
'metrics': {'package': 'fn-aws-iam', 'timestamp': '2020-02-19
12:53:48', 'package_version': '1.0.0',
'host': 'myhost.ibm.com', 'version': '1.0',
'execution_time_ms': 3023},
'success': True,
'content': 'OK',
'raw': '"OK"',
'reason': None,
'version': '1.0'
}
```

► Example Pre-Process Script:

```
inputs.aws_iam_access_key_id = row.AccessKeyId
inputs.aws_iam_user_name = row.UserName
inputs.aws_iam_status = "Inactive"
```

► Example Post-Process Script:

```
# Globals
# List of fields in datatable for fn_aws_iam_delete_access_keys script
DATA_TBL_FIELDS = ["AccessKeyIds"]
FN_NAME = "fn_aws_iam_update_access_key"
```

```

WF_NAME = "Deactivate Access Key"
# Processing
CONTENT = results.content
INPUTS = results.inputs

def main():
    note_text = u''
    if CONTENT:
        if CONTENT == "OK":
            note_text = u"AWS IAM Integration: Workflow <b>{0}</b>: The
Access Key Id <b>{1}</b> was deactivated " \
                u"for user <b>{2}</b> for Resilient function <b>
{3}</b>." \
                    .format(WF_NAME, INPUTS["aws_iam_access_key_id"],
INPUTS["aws_iam_user_name"], FN_NAME)
            row.Status = "Inactive"
        elif CONTENT == u"NoSuchEntity":
            note_text = u"AWS IAM Integration: Workflow <b>{0}</b>: The
Access Key Id <b>{1}</b> Not found " \
                u"for user <b>{2}</b> for Resilient function <b>
{3}</b>." \
                    .format(WF_NAME, INPUTS["aws_iam_access_key_id"],
INPUTS["aws_iam_user_name"], FN_NAME)
            row.Status = "NoSuchEntity"
        else:
            note_text += u"AWS IAM Integration: Workflow <b>{0}</b>: There
were no results returned for " \
                u"access key id <b>{1}</b> access key Resilient
function <b>{2}</b>." \
                    .format(WF_NAME, INPUTS["aws_iam_access_keys"], 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,AKIA4EQBBG2YGLNUP064	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	—	—

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

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?

3. Press Execute to invoke the **Example: AWS IAM: Delete Access Keys** workflow, which calls the **AWS IAM: Delete Access Keys** function.

On successful completion of the workflow, the **Access key ids** field of the **AWS IAM Users** data table is updated for the selected user.

AWS IAM Users

iam_test_user_1

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

The 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?

Cancel Execute

3. Press Execute to invoke the **Example: AWS IAM: Delete Access Key** workflow, which calls the **AWS IAM: Delete Access Keys** function.

On successful completion of the workflow, the **Status** field of the **AWS IAM Access Keys** data table is transitioned to **Deleted**.

AWS IAM Access Keys

					iam_test_user_6		Print	Export
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	

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



- Press Execute to invoke the **Example: AWS IAM: Delete Access Key For Artifact** workflow, which calls the **AWS IAM: Delete Access Keys** function.

On successful completion of the workflow, the artifact description is 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	<p>AWS IAM access key detected for AWS IAM query by function 'scr_aws_iam_add_access_key_as_artifact' for AWS IAM.</p> <p>=====</p> <p>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'.</p> <p>=====</p>
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 = {'version': '1.0', 'success': True, 'reason': None,
           'content': [{'AccessKeyId': 'AKIA4EQBBG2YFAXXIG6M', 'Status':
                        'OK'}],
           'raw': '[{"AccessKeyId": "AKIA4EQBBG2YFAXXIG6M", "Status":
                    "OK"}]',
           'inputs': {'aws_iam_user_name': 'iam_test_User_1',
                      'aws_iam_access_keys': 'AKIA4EQBBG2YFAXXIG6M'},
           'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
                      'package_version': '1.0.0',
                      'host': 'myhost.ibm.com', 'execution_time_ms': 752,
                      'timestamp': '2020-01-16 13:47:07'}
          }

```

► 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 ##
# 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 Key"
# Processing
CONTENT = results.content
INPUTS = results.inputs
EXECUTION_DATE = results["metrics"]["timestamp"]

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>: The
            Access Key Id <b>{1}</b> was deleted " \

```



```

                                "for user <b>{2}</b> for Resilient function <b>{3}
</b>." \
                                .format(WF_NAME, ','.join(deleted_keys),
INPUTS["aws_iam_user_name"], FN_NAME)
                                if no_such_entity:
                                    note_text = "AWS IAM Integration: : Workflow <b>{0}</b>:
Access keyId id <b>{1}</b> does not exist " \
                                "for user <b>{2}</b> for Resilient function <b>{3}
</b>." \
                                .format(WF_NAME, ','.join(no_such_entity_keys),
INPUTS["aws_iam_user_name"], FN_NAME)
                                    row.Status = "Deleted"
                                    row.query_execution_date = EXECUTION_DATE
                                else:
                                    note_text += "AWS IAM Integration: Workflow <b>{0}</b>: There were
no results returned for " \
                                    "access key id <b>{1}</b> access key Resilient function <b>{2}
</b>." \
                                    .format(WF_NAME, INPUTS["aws_iam_access_keys"], FN_NAME)

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

```

Function - AWS IAM: List User Groups

Gets the IAM groups that include 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: Refresh User](#), [Example: AWS IAM: Add User To Group](#), [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: Refresh User](#), 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: Refresh User](#).

1. Open an incident and select the row of data table [AWS IAM Users](#) corresponding to the user who needs to be added to a group.
2. From the selected row's actions menu, select [Example: AWS IAM: Refresh User](#).

AWS IAM Users

iam_test_user_10

Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 17:01:04	iam_test_user_10	Yes	AKIA4EQBBG2YGRLFWXAP	—	deny_all,AWSDe	nyAll	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

Displaying 1 - 1 of 1 (filtered from 14 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: Get Access Keys
Example: AWS IAM: Refresh User

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name
2020-03-11 17:01:04	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—	N/A
2020-03-11 17:01:04	AKIA4EQBBG2YP4S4SOEV	adminuser	2019-11-04 11:33:33	Active	Yes	2020-03-11 16:59:00	iam

This invokes the **Example: AWS IAM: Refresh User** workflow, which calls the **AWS IAM: List User Groups** function.

On successful completion of the workflow, the **Groups** field of the **AWS IAM Users** data table is updated for the selected user.

AWS IAM Users

iam_test_user_10

Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 16:54:52	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4MROM	denyall_group	deny_all,AWSDe	nyAll	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

► Inputs:

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

► Outputs:

```
results = {'version': '1.0', 'success': True, 'reason': None,
          'content': [{'Path': '/', 'GroupName': 'null_group', 'GroupId':
'AGPA4EQBBG2YAVPJATCNZ',
'Arn': 'arn:aws:iam::834299573936:group/null_group',
'CreateDate': '2019-12-04 12:31:47'},
{'Path': '/', 'GroupName': 'denyall_group',
'GroupId': 'AGPA4EQBBG2YPUAIHTA3E',
'Arn':
'arn:aws:iam::834299573936:group/denyall_group', 'CreateDate': '2019-11-29
15:49:34'},
{'Path': '/', 'GroupName': 'myS3group', 'GroupId':
'AGPAIH4V2XCXOME6HWCPQ',
'Arn': 'arn:aws:iam::834299573936:group/myS3group',
'CreateDate': '2017-05-29 20:41:50'}],
'raw': '[{"Path": "/", "GroupName": "null_group", "GroupId":
"AGPA4EQBBG2YAVPJATCNZ", "
"Arn": "arn:aws:iam::834299573936:group/null_group",
"CreateDate": "2019-12-04 12:31:47"}, {"
"Path": "/", "GroupName": "denyall_group", "GroupId":
"AGPA4EQBBG2YPUAIHTA3E", "
"Arn": "arn:aws:iam::834299573936:group/denyall_group",
"CreateDate": "2019-11-29 15:49:34"}, {"
"Path": "/", "GroupName": "myS3group", "GroupId":
"AGPAIH4V2XCXOME6HWCPQ",
"Arn": "arn:aws:iam::834299573936:group/myS3group",
"CreateDate": "2017-05-29 20:41:50"}]}
```

```

        '{"Path": "/", "GroupName": "myS3group", "GroupId":
"AGPAIH4V2XCX0ME6HWCPQ",' \
        '"Arn": "arn:aws:iam::834299573936:group/myS3group",
"CreateDate": "2017-05-29 20:41:50"}]' ,
        'inputs': {'aws_iam_user_name': 'iam_test_user_1'},
        'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0', 'host': 'myhost.ie.ibm.com',
        'execution_time_ms': 944, 'timestamp': '2020-03-16
15:43:21'
        }
    }
}

```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```

## AWS IAM - fn_aws_iam_list_user_groups script ##
# 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 = "Refresh User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
note_text = ''

def main():

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

    if __name__ == "__main__":
        main()

```

Function - AWS IAM: Add User To Groups

Adds 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 an 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 the 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_10

id	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
3-1252	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4MROM	—	deny_all,AWSDe	nyAll	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name
2020-03-11 17:01:04	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—	N/A
2020-03-11 17:01:04	AKIA4EQBBG2YP4S4SOEV	adminuser	2019-11-04 11:33:33	Active	Yes	2020-03-11 16:52:00	iam

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: Get Access Keys
 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
 null_group
 myS3group
 not_exists

This invokes the [Example: AWS IAM: Add User To Group](#) workflow, which calls the [AWS IAM: Add User To Groups](#) function.

On successful completion of the workflow, the `Groups` field of the [AWS IAM Users](#) data table is updated for the selected user.

AWS IAM Users

iam_test_user_10

id	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
3-1254	iam_test_user_10	Yes	AKIA4EQBBG2YGRFLWXAP	denyall_group	deny_all,AWSDe	nyAll	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

► Inputs:

Name	Type	Required	Example	Tooltip
------	------	----------	---------	---------

Name	Type	Required	Example	Tooltip
<code>aws_iam_group_names</code>	text	Yes	–	Comma-separated list of AWS IAM group names.
<code>aws_iam_user_name</code>	text	Yes	AWS IAM user name	AWS IAM user name.

► Outputs:

```
{'version': '1.0', 'success': True, 'reason': None,
  'content': [{'GroupName': 'denyall_group', 'Status': 'OK'}],
  'raw': '[{"GroupName": "denyall_group", "Status": "OK"}]',
  'inputs': {'aws_iam_user_name': 'iam_test_user_1',
    'aws_iam_group_names': 'denyall_group'},
  'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
    'package_version': '1.0.0',
    'host': 'myhost.ibm.com', 'execution_time_ms': 4303,
    'timestamp': '2020-03-16 15:43:17'
  }
}
```

► 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 ##
# 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>: User
<b>{1}</b> added to group <b>{2}</b> " \
            "for Resilient function <b>{3}</b>." \
            .format(WF_NAME, INPUTS["aws_iam_user_name"], ",
".join(str(i) for i in added_groups), FN_NAME)
        note_text += "<br>Refreshing data table <b>{0}</b> row for
user <b>{1}</b> with updated group data." \
            .format("AWS IAM Users", INPUTS["aws_iam_user_name"])
    if no_such_entity:
        note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: The
group(s) <b>{1}</b> " \
            "did not exist for user <b>{2}</b> for Resilient
function <b>{3}</b>." \
            .format(WF_NAME, ", ".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: Remove User From Groups

Removes the specified IAM user from the specified groups. Group names is 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 the 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_10 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 16:52	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4MROM	denyall_group,myS3group	deny_all,AWSDe	nyAll	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date
2020-03-11 17:01:04	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—
2020-03-11 17:01:04	AKIA4EQBBG2YP4S4SOEV	adminuser	2019-11-04 11:33:33	Active	Yes	2020-03-11 16:52:00

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: Get Access Keys
 Example: AWS IAM: Refresh User
 Example: AWS IAM: Remove User From All Groups

The 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?

Cancel Execute

3. Press Execute to invoke the **Example: AWS IAM: Remove User From All Groups** workflow, which calls the **AWS IAM: Remove User From Groups** function.

On successful completion of the workflow, the **Groups** field of the **AWS IAM Users** data table is updated to an empty value for the selected user.

AWS IAM Users

iam_test_user_10 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 16:54:52	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4MROM	—	deny_all,AWSDe	nyAll	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

► 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 = {
    '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': 'OK'}],
    'raw': '[{"PolicyArn": "arn:aws:iam::aws:policy/AWSDenyAll",
"Status": "OK"},' \
          '{"PolicyArn": "arn:aws:iam::aws:policy/AWSDenyAll_2",
"Status": "OK"}]',
    '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'
                }
}

```

► 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 ##
# 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>: The
user <b>{1}</b> was removed from the " \
                "following groups <b>{2}</b> for Resilient
function <b>{3}</b>." \
                .format(WF_NAME, INPUTS["aws_iam_user_name"], ",
".join(str(i) for i in added_groups), 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: List User Policies

Gets 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: Refresh User](#), [Example: AWS IAM: Attach User Policy](#), [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: Refresh User](#), 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: Refresh User](#).

1. Open an incident and select the row of data table [AWS IAM Users](#) corresponding to the user who needs to have a policy attached.
2. From the selected rows action menu, select [Example: AWS IAM: Refresh User](#).

AWS IAM Users

iam_test_user_10 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 17:00:00	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4 MROM	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name
2020-03-11 17:01:04	AKIA4EQBBG2Y03VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—	N/A

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: Get Access Keys
 Example: AWS IAM: Refresh User

This invokes the **Example: AWS IAM: Refresh User** workflow, which calls the **AWS IAM: List User Policies** function.

On successful completion of the workflow, the **Policies** field of the **AWS IAM Users** data table is updated for the selected user.

AWS IAM Users

iam_test_user_10 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 19:35:04	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4 MROM	—	deny_all,AWSDe nyAll	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

► Inputs:

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

► Outputs:

```
results = {
  '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'
  }
}
```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```
## AWS IAM - fn_aws_iam_list_user_policies script ##
# Globals
# List of fields in datatable fn_aws_iam_list_user_groups script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_list_user_policies"
WF_NAME = "Refresh User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
note_text = ''

def main():
    note_text = ''
    if CONTENT:
        policy_names = []
        for pol in CONTENT:
            if pol["PolicyName"] is not None:
                policy_names.append( pol["PolicyName"])
        row.Policies = ",".join(policy_names)
    else:
        row.Policies = ""

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

Function - AWS IAM: Attach User policies

Attaches 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`, is 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**.

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

AWS IAM Users

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 17:01:04	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4 MROM	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name
2020-03-11 17:01:04	AKIA4EQBBG2YO3VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—	N/A

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: Get Access Keys
 Example: AWS IAM: Refresh User

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

Example: AWS IAM: Attach User Policy

Policy name

AWSDenyAll
 AWSDenyAll
 deny_all
 not_exists

Cancel Execute

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

On successful completion of the workflow, the **Policies** field of the **AWS IAM Users** data table is updated for the selected user.

AWS IAM Users

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status
2020-03-12 19:35:04	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4 MROM	—	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active

► 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 = {
    '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'
                }
}
```

► 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 ##
# 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>: The
Policy <b>{1}</b> was attached to user " \
            "<b>{2}</b> for Resilient function <b>{3}</b>." \
            .format(WF_NAME, ", ".join(str(i) for i in
added_policies), INPUTS["aws_iam_user_name"], FN_NAME)
        note_text += "<br>Refreshing data table <b>{0}</b> row for
user <b>{1}</b> with updated policy data." \
            .format("AWS IAM Users", INPUTS["aws_iam_user_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: Detach User policies

Removes 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 have embedded inline policies, which the function also deletes. One of the parameters, `aws_iam_policy_names` or `aws_iam_arns`, is 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

iam_test_user_10 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status	
2020-03-12 17:01:04	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4MROM	—	deny_all,AWSDe	nyAll	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active	...

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name
2020-03-11 17:01:04	AKIA4EQBBG2Y03VWDSN6	adminuser	2019-10-31 16:23:08	Active	—	—	N/A
2020-03-11 17:01:04	AKIA4EQBBG2YP4S4SOEV	adminuser	2019-11-04 11:33:33	Active	Yes	2020-03-11 17:01:04	iam

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: Get Access Keys
 Example: AWS IAM: Refresh User

The 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?

Cancel Execute

3. Press Execute to invoke the **Example: AWS IAM: Detach All User Policies** workflow, which calls the **AWS IAM: Detach User policies** function.

On successful completion of the workflow, the **Policies** field of the **AWS IAM Users** data table is updated to an empty value for the selected user.

AWS IAM Users

iam_test_user_10 Print Export

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status	
2020-03-12 17:01:04	iam_test_user_10	Yes	AKIA4EQBBG2YC3T4MROM	—	—	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active	...

► 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.

Name	Type	Required	Example	Tooltip
<code>aws_iam_user_name</code>	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 = {
    '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'
}
}
```

► 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 ##
# 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
detached_policies = []
no_such_entity_policies = []
if CONTENT:
    for pol_stat in CONTENT:
        if pol_stat["Status"] == "OK":
            added += 1
            detached_policies.append(pol_stat["PolicyName"])
        else:
            no_such_entity += 1
            no_such_entity_policies.append(pol_stat["PolicyName"])
    if detached_policies:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: The Policies <b>{1}</b> were detached and/or deleted " \
            "for user <b>{2}</b> for Resilient function <b>{3}</b>." \
            .format(WF_NAME, ", ".join(str(i) for i in detached_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 SSH Public Keys

List the SSH public keys associated with an IAM user. Parameter `aws_iam_user_name` is an IAM user name.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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 User](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field 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 data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

► Inputs:

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': [{ 'UserName': 'iam_test_user', 'SSHPublicKeyId':
'APKA4EQBBG2YCG0GDY5G',
                  'Status': 'Active', 'UploadDate': '2020-02-25
11:05:17'
                }
    ],
    'raw': '[{"UserName": "iam_test_user_10", "SSHPublicKeyId":
"APKA4EQBBG2YCG0GDY5G", "\
                "Status": "Active", "UploadDate": "2020-02-25
11:05:17"}]',
    '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': 657, 'timestamp': '2020-02-25
16:11:28'
    }
}
```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```
## AWS IAM - fn_aws_iam_list_ssh_public_keys script ##
# Globals
# List of fields in datatable fn_aws_iam_list_ssh_public_keys script
DATA_TBL_FIELDS = ["SSHPublicKeyIds"]
FN_NAME = "fn_aws_iam_list_ssh_public_keys"
WF_NAME = "Delete User"
# Processing
```

```

CONTENT = results.content
INPUTS = results.inputs
DEBUG_SCRIPT=False

def main():
    note_text = ''
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>{1}</b> 'SSH Public keys' 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 sshk_id in CONTENT:
            if sshk_id["SSHPublicKeyId"] is not None:
                workflow.addProperty("has_ssh_public_keys", {})
                break
        else:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>no</b> 'SSH Public keys' result(s) returned for " \
                "user <b>{1}</b> for Resilient function <b>{2}</b>." \
                .format(WF_NAME, INPUTS["aws_iam_user_name"], FN_NAME)
    if DEBUG_SCRIPT:
        incident.addNote(helper.createRichText(note_text))

if __name__ == "__main__":
    main()

```

Function - AWS IAM: Delete SSH Public Keys

Delete Secure Shell (SSH) public keys associated with the specified IAM user. Parameter `aws_iam_user_name` is an IAM user name. Parameter `aws_iam_ssh_key_ids` is a comma-separated list of SSH public key IDs.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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_ssh_key_ids` is mapped to SSH key IDs retrieved from the previous step (c.f. [AWS IAM: List SSH Public Keys](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field for the function:

- `aws_iam_user_name` is mapped to an artifact value for artifact of type [AWS IAM User Name](#).

- `aws_iam_ssh_key_ids` is mapped to SSH key IDs retrieved from the previous step (c.f. [AWS IAM: List SSH Public Keys](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_ssh_key_ids</code>	text	No	–	–
<code>aws_iam_user_name</code>	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```
results = {
    'version': '1.0', 'success': True, 'reason': None,
    'content': [{ 'SSHPublicKeyId': 'APKA4EQBBG2YCG0GDY5G', 'Status':
'OK' }
                { 'SSHPublicKeyId': 'APKA4EQBBG2YCG0GDY5G', 'Status':
'NoSuchEntity' } ],
    'raw': '[{"SSHPublicKeyId": "APKA4EQBBG2YCG0GDY5G", "Status":
"OK"}, \
          {"SSHPublicKeyId": "APKA4EQBBG2YCG0GDY5G", "Status":
"NoSuchEntity"}]',
    'inputs': { 'aws_iam_ssh_keys_ids': 'APKA4EQBBG2YCG0GDY5G',
'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': 790, 'timestamp': '2019-11-29
12:18:30'
    }
}
```

► Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
content = workflow.properties.list_ssh_keys_results.content
ssh_key_ids = []
for ssh_key_id in content:
    if ssh_key_id["SSHPublicKeyId"] is not None:
        ssh_key_ids.append(ssh_key_id["SSHPublicKeyId"])
inputs.aws_iam_ssh_key_ids = ",".join(ssh_key_ids)
```

► Example Post-Process Script:

```
## AWS IAM - fn_aws_iam_delete_ssh_keys script ##
# Globals
```

```

# List of fields in datatable for fn_aws_iam_delete_ssh_keys script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_delete_ssh_keys"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
DEBUG_SCRIPT = False

def main():
    note_text = ''
    deleted = 0
    no_such_entity = 0
    deleted_keys = []
    no_such_entity_keys = []
    if CONTENT:
        for grp_stat in CONTENT:
            if grp_stat["Status"] == "OK":
                deleted += 1
                deleted_keys.append(grp_stat["GroupName"])
            else:
                no_such_entity += 1
                no_such_entity_keys.append(grp_stat["GroupName"])
        if deleted_keys:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> 'SSH Public keys' <b>{2}</b> removed " \
                "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> 'SSH Public keys' <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)
            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)
        if DEBUG_SCRIPT:
            incident.addNote(helper.createRichText(note_text))
    if __name__ == "__main__":
        main()

```

Function - AWS IAM: List Service Specific Credentials

List the service-specific credentials associated with an IAM user. Parameter `aws_iam_user_name` is an IAM user name.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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 User](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field 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 data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_user_name</code>	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```
results = {'version': '1.0', 'success': True, 'reason': None,
          'content': [{'UserName': 'iam_test_user', 'Status': 'Active',
'ServiceUserName': 'iam_test_user-at-834299573936',
          'CreateDate': '2020-02-25 10:43:24',
'ServiceSpecificCredentialId': 'ACCA4EQBBG2YH6NR76SCQ',
          'ServiceName': 'codecommit.amazonaws.com'
},
          {'UserName': 'iam_test_user', 'Status': 'Active',
'ServiceUserName': 'iam_test_user_10-at-834299573936',
          'CreateDate': '2020-02-26 11:50:52',
'ServiceSpecificCredentialId': 'ACCA4EQBBG2YG0LHZWY7L',
          'ServiceName': 'cassandra.amazonaws.com'}]},
          'raw': '[{"UserName": "iam_test_user", "Status": "Active",
"ServiceUserName": "iam_test_user_10-at-834299573936", "\
          "CreateDate": "2020-02-25 10:43:24",
"ServiceSpecificCredentialId": "ACCA4EQBBG2YH6NR76SCQ", "\
          "ServiceName": "codecommit.amazonaws.com"},
{"UserName": "iam_test_user_10", "Status": "Active", "\
          "ServiceUserName": "iam_test_user_10-at-834299573936",
"CreateDate": "2020-02-26 11:50:52", "\
          "ServiceSpecificCredentialId":
"ACCA4EQBBG2YG0LHZWY7L", "ServiceName": "cassandra.amazonaws.com"}]',
          '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': 982, 'timestamp': '2020-02-26
11:56:51'
                }
    }

```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```

## AWS IAM - fn_aws_iam_list_service_specific_credentials script ##
# Globals
# List of fields in datatable fn_aws_iam_list_service_specific_credentials
script
DATA_TBL_FIELDS = ["ServiceSpecificCredentialIds"]
FN_NAME = "fn_aws_iam_list_service_specific_credentials"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
DEBUG_SCRIPT = False

def main():
    note_text = ''
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>{1}</b> 'Service specific credentials' returned for user " \
                    "<b>{2}</b> for Resilient function <b>{3}</b>." \
                    .format(WF_NAME, len(CONTENT), INPUTS["aws_iam_user_name"],
FN_NAME)
        for ssc_id in CONTENT:
            if ssc_id["ServiceSpecificCredentialId"] is not None:
                workflow.addProperty("has_srv_creds", {})
                break
        else:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>no</b> 'Service specific credentials' result(s) returned for " \
                    "user <b>{1}</b> for Resilient function <b>{2}</b>." \
                    .format(WF_NAME, INPUTS["aws_iam_user_name"], FN_NAME)
        if DEBUG_SCRIPT:
            incident.addNote(helper.createRichText(note_text))

if __name__ == "__main__":
    main()

```

Function - AWS IAM: Delete Service Specific Credentials

Delete service-specific credentials associated with the specified IAM user. Parameter `aws_iam_user_name` is an IAM user name. Parameter `aws_iam_ssc_ids` is a comma-separated list of service-specific credential ids.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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_ssc_ids` is mapped to service specific credential IDs retrieved from the previous step (c.f. [AWS IAM: List Service Specific Credentials](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User](#). For more information on this workflow reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field for the function:

- `aws_iam_user_name` is mapped to an artifact value for artifact of type [AWS IAM User Name](#).
- `aws_iam_ssc_ids` is mapped to service specific credential IDs retrieved from the previous step (c.f. [AWS IAM: List Service Specific Credentials](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_ssc_ids</code>	text	No	–	–
<code>aws_iam_user_name</code>	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```
results = {
    'version': '1.0', 'success': True, 'reason': None,
    'content': [{ 'ServiceSpecificCredentialId':
'ACCA4EQBBG2YH6NR76SCQ', 'Status': 'OK' }
    { 'ServiceSpecificCredentialId':
'ACCA4EQBBG2YH6NR76SCQ', 'Status': 'NoSuchEntity' }],
    'raw': '[{"ServiceSpecificCredentialId": "ACCA4EQBBG2YH6NR76SCQ",
"Status": "OK"}, \
    {"ServiceSpecificCredentialId": "ACCA4EQBBG2YH6NR76SCQ",
"Status": "NoSuchEntity"}]',
    'inputs': { 'aws_iam_ssc_ids': 'ACCA4EQBBG2YH6NR76SCQ',
'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': 790, 'timestamp': '2019-11-29
12:18:30'
    }
}

```

► Example Pre-Process Script:

```

inputs.aws_iam_user_name = row.UserName
content = workflow.properties.list_srv_specific_creds_results.content
srv_specific_cred_ids = []
for ssc_id in content:
    if ssc_id["ServiceSpecificCredentialId"] is not None:

srv_specific_cred_ids.append(ssc_id["ServiceSpecificCredentialId"])
inputs.aws_iam_ssc_ids = ",".join(srv_specific_cred_ids)

```

► Example Post-Process Script:

```

## AWS IAM - fn_aws_iam_delete_ss_creds script ##
# Globals
# List of fields in datatable for fn_aws_iam_delete_ss_creds script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_delete_ss_creds"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
DEBUG_SCRIPT = False

def main():
    note_text = ''
    deleted = 0
    no_such_entity = 0
    deleted_creds = []
    no_such_entity_creds = []
    if CONTENT:
        for grp_stat in CONTENT:
            if grp_stat["Status"] == "OK":
                deleted += 1
                deleted_creds.append(grp_stat["GroupName"])
            else:
                no_such_entity += 1
                no_such_entity_creds.append(grp_stat["GroupName"])
        if deleted_creds:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> 'Service specific credentials' <b>{2}</b> removed " \
                "for user <b>{3}</b> for Resilient function <b>{4}

```

```

</b>." \
        .format(WF_NAME, len(deleted_creds), ", ".join(str(i) for
i in deleted_creds), 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> 'Service specific credentials' <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_creds), ",
".join(str(i) for i in no_such_entity_creds), 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)
        if DEBUG_SCRIPT:
            incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()

```

Function - AWS IAM: List Signing Certificates

List the signing certificates associated with an IAM user. Parameter `aws_iam_user_name` is an IAM user name.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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 User](#). For more information on this workflow reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field 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 data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_user_name</code>	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```

results = {'version': '1.0', 'success': True, 'reason': None,
          'content': [{'UserName': 'iam_test_user', 'CertificateId':
'WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD',
          'CertificateBody': '-----BEGIN CERTIFICATE-----
\nMIID...Apg=\n-----END CERTIFICATE-----',
          'Status': 'Active', 'UploadDate': '2020-02-26
12:25:27'}]},
          'raw': '[{"UserName": "iam_test_user", "CertificateId":
"WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD", "CertificateBody": '
          '-----BEGIN CERTIFICATE-----\n\nMIID...Apg=\n-----END
CERTIFICATE-----', "Status": "Active",'
          '"UploadDate": "2020-02-26 12:25:27"}]'}],
          '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': 729, 'timestamp': '2020-02-26
12:33:57'
          }
}

```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```

## AWS IAM - fn_aws_iam_list_signing_certificates script ##
# Example result:
"""
Result: {'version': '1.0', 'success': True, 'reason': None,
        'content': [{'UserName': 'iam_test_user', 'CertificateId':
'WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD',
        'CertificateBody': '-----BEGIN CERTIFICATE-----
\nMIID...Apg=\n-----END CERTIFICATE-----',
        'Status': 'Active', 'UploadDate': '2020-02-26
12:25:27'}]},
        'raw': '[{"UserName": "iam_test_user", "CertificateId":
"WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD", "CertificateBody":
        "-----BEGIN CERTIFICATE-----\n\nMIID...Apg=\n-----END
CERTIFICATE-----', "Status": "Active", "UploadDate": "2020-02-26
12:25:27"}]'}],
        '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': 729, 'timestamp': '2020-02-26
12:33:57'
        }
}

```

```

####
# Globals
# List of fields in datatable fn_aws_iam_list_signing_certificates script
DATA_TBL_FIELDS = ["CertificateIds"]
FN_NAME = "fn_aws_iam_list_signing_certificates"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
DEBUG_SCRIPT=False

def main():
    note_text = ''
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There were
<b>{1}</b> 'Signing Certificates' returned for user " \
            "<b>{2}</b> for Resilient function <b>{3}</b>." \
            .format(WF_NAME, len(CONTENT), INPUTS["aws_iam_user_name"],
FN_NAME)
        for scert_id in CONTENT:
            if scert_id["CertificateId"] is not None:
                workflow.addProperty("has_sign_certs", {})
                break
    else:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
<b>no</b> 'Signing Certificates' result(s) returned for " \
            "user <b>{1}</b> for Resilient function <b>{2}</b>." \
            .format(WF_NAME, INPUTS["aws_iam_user_name"], FN_NAME)
    if DEBUG_SCRIPT:
        incident.addNote(helper.createRichText(note_text))

if __name__ == "__main__":
    main()

```

Function - AWS IAM: Delete Signing Certificates

Delete signing certificates associated with the specified IAM user. Parameter `aws_iam_user_name` is an IAM user name. Parameter `aws_iam_sign_cert_ids` is a comma-separated list of signing certificate ids.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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_sign_cert_ids` is mapped to signing certificate IDs retrieved from the previous step (c.f. [AWS IAM: List Signing Certificates](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field for the function:

- `aws_iam_user_name` is mapped to an artifact value for artifact of type [AWS IAM User Name](#).
- `aws_iam_sign_cert_ids` is mapped to signing certificate IDs retrieved from the previous step (c.f. [AWS IAM: List Signing Certificates](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_sign_cert_ids</code>	text	No	–	–
<code>aws_iam_user_name</code>	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```
results = {
    'version': '1.0', 'success': True, 'reason': None,
    'content': [{ 'CertificateId':
'WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD', 'Status': 'OK'},
                { 'CertificateId':
'WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD', 'Status': 'NoSuchEntity'}],
    'raw': '[{ "CertificateId": "WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD",
"Status": "OK"},'
            { "CertificateId": "WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD",
'Status': "NoSuchEntity"}]',
    'inputs': { 'aws_iam_sign_cert_ids':
'WM6U3NNR5JH3A0TNJY44CUI6I6EYXTLD', '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': 790, 'timestamp': '2019-11-29
12:18:30'
    }
}
```

► Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
content = workflow.properties.list_signing_certs_results.content
sign_cert_ids = []
for scert_id in content:
    if scert_id["CertificateId"] is not None:
        sign_cert_ids.append(scert_id["CertificateId"])
inputs.aws_iam_sign_cert_ids = ",".join(sign_cert_ids)
```

► Example Post-Process Script:

```

## AWS IAM - fn_aws_iam_list_signing_certs script ##

# Globals
# List of fields in datatable for fn_aws_iam_list_signing_certs script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_list_signing_certs"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
DEBUG_SCRIPT = False

def main():
    note_text = ''
    deleted = 0
    no_such_entity = 0
    deleted_certs = []
    no_such_entity_certs = []
    if CONTENT:
        for grp_stat in CONTENT:
            if grp_stat["Status"] == "OK":
                deleted += 1
                deleted_certs.append(grp_stat["GroupName"])
            else:
                no_such_entity += 1
                no_such_entity_certs.append(grp_stat["GroupName"])
        if deleted_certs:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> 'Signing Certificates' <b>{2}</b> removed " \
                "for user <b>{3}</b> for Resilient function <b>{4}</b>." \
                .format(WF_NAME, len(deleted_certs), ", ".join(str(i) for
i in deleted_certs), 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> 'Signing Certificates' <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_certs), ",
".join(str(i) for i in no_such_entity_certs), 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)
        if DEBUG_SCRIPT:
            incident.addNote(helper.createRichText(note_text))
    if __name__ == "__main__":
        main()

```

Function - AWS IAM: List MFA Devices

List the MFA devices associated with an IAM user. Also determine which of the associated MFA devices is a virtual device. Parameter `aws_iam_user_name` is an IAM user name.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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 User](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field 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 data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_user_name</code>	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```
results = {'version': '1.0', 'success': True, 'reason': None,
          'content': [{'UserName': 'iam_test_user', 'SerialNumber':
'arn:aws:iam::834299573936:mfa/iam_test_user',
          'EnableDate': '2020-02-26 16:55:05', 'is_virtual':
True}],
          'raw': '[{"UserName": "iam_test_user_10", "SerialNumber":
"arn:aws:iam::834299573936:mfa/iam_test_user_10",
          "EnableDate": "2020-02-26 16:55:05", "is_virtual":
True}]',
          'inputs': {'aws_iam_user_name': 'iam_test_user_10'},
          'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0', 'host': 'myhost.ie.ibm.com',
          'execution_time_ms': 5644, 'timestamp': '2020-02-26
17:37:48'
          }
}
```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```
## AWS IAM - fn_aws_iam_list_mfa_devices script ##
# Globals
# List of fields in datatable fn_aws_iam_list_mfa_devices script
DATA_TBL_FIELDS = ["SerialNumbers"]
FN_NAME = "fn_aws_iam_list_mfa_devices"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
DEBUG_SCRIPT=False

def main():
    note_text = ''
    if CONTENT:
        note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There were
        <b>{1}</b> 'Active NFA devices' returned for user " \
            "<b>{2}</b> for Resilient function <b>{3}</b>." \
            .format(WF_NAME, len(CONTENT), INPUTS["aws_iam_user_name"],
            FN_NAME)
        for mfa_ser_num in CONTENT:
            if mfa_ser_num["SerialNumber"] is not None:
                workflow.addProperty("has_active_mfa", {})
            if mfa_ser_num.get("is_virtual", None):
                workflow.addProperty("is_virtual_mfa", {})
        else:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There was
            <b>no</b> 'Active NFA devices' result(s) returned for " \
                "user <b>{1}</b> for Resilient function <b>{2}</b>." \
                .format(WF_NAME, INPUTS["aws_iam_user_name"], FN_NAME)
        if DEBUG_SCRIPT:
            incident.addNote(helper.createRichText(note_text))

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

Function - AWS IAM: Deactivate MFA Devices

Deactivate MFA devices and remove it from association with the user name for which it was originally enabled. Parameter `aws_iam_user_name` is an IAM user name. Parameter `aws_iam_mfa_serial_numbers` is a comma-separated list of IAM MFA serial numbers or arns.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field 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_mfa_serial_numbers` is mapped to MFA serial numbers retrieved from the previous step (c.f. [AWS IAM: List MFA Devices](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field for the function:

- `aws_iam_user_name` is mapped to an artifact value for artifact of type [AWS IAM User Name](#).
- `aws_iam_mfa_serial_numbers` is mapped to MFA serial numbers retrieved from the previous step (c.f. [AWS IAM: List MFA Devices](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_mfa_serial_nums</code>	text	No	–	–
<code>aws_iam_user_name</code>	text	No	AWS IAM user name	AWS IAM user name.

► Outputs:

```
results = {
  'version': '1.0', 'success': True, 'reason': None,
  'content': [{ 'SerialNumber':
'arn:aws:iam::834299573936:mfa/iam_test_user', 'Status': 'OK'}],
  'raw': '[{"SerialNumber":
"arn:aws:iam::834299573936:mfa/iam_test_user", "Status": "OK"}]',
  'inputs': { 'aws_iam_mfa_serial_nums':
'arn:aws:iam::834299573936:mfa/iam_test_user',
              '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': 790, 'timestamp': '2019-11-29
12:18:30'
              }
}
```

► Example Pre-Process Script:

```
inputs.aws_iam_user_name = row.UserName
content = workflow.properties.list_mfa_devices_results.content
mfa_serial_nums = []
```

```

for mfa_ser_num in content:
    if mfa_ser_num["SerialNumber"] is not None:
        mfa_serial_nums.append(mfa_ser_num["SerialNumber"])
inputs.aws_iam_mfa_serial_nums = ",".join(mfa_serial_nums)

```

► Example Post-Process Script:

```

## AWS IAM - fn_aws_iam_deactivate_mfa_devices script ##
# Globals
# List of fields in datatable for fn_aws_iam_deactivate_mfa_devices
script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_deactivate_mfa_devices"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
DEBUG_SCRIPT = False

def main():
    note_text = ''
    deactivated = 0
    no_such_entity = 0
    deactivated_mfas = []
    no_such_entity_mfas = []
    if CONTENT:
        for grp_stat in CONTENT:
            if grp_stat["Status"] == "OK":
                deactivated += 1
                deactivated_mfas.append(grp_stat["GroupName"])
            else:
                no_such_entity += 1
                no_such_entity_mfas.append(grp_stat["GroupName"])
        if deactivated_mfas:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> 'MFA devices' <b>{2}</b> de-activated " \
                "for user <b>{3}</b> for Resilient function <b>{4}</b>." \
                .format(WF_NAME, len(deactivated_mfas), ", ".join(str(i)
for i in deactivated_mfas), 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> 'MFA devices' <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_mfas), ",
".join(str(i) for i in no_such_entity_mfas), 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)
    if DEBUG_SCRIPT:
        incident.addNote(helper.createRichText(note_text))
    if __name__ == "__main__":
        main()

```

Function - AWS IAM: Delete Virtual MFA Devices

Delete a virtual MFA device. Parameter `aws_iam_mfa_serial_numbers` is a comma-separated list of IAM MFA serial numbers or arns.

Note: You must deactivate a user's virtual MFA device before you can delete it.

Example workflows that use this Resilient function include [Example: AWS IAM: Delete User](#) or [Example: AWS IAM: Delete User For Artifact](#).

The workflow, [Example: AWS IAM: Delete User](#), sets the following input field for the function:

- `aws_iam_mfa_serial_numbers` is mapped to MFA serial numbers retrieved a previous step (c.f. [AWS IAM: List MFA Devices](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

The workflow, [Example: AWS IAM: Delete User For Artifact](#), sets the following input field for the function:

- `aws_iam_mfa_serial_numbers` is mapped to MFA serial numbers retrieved a previous step (c.f. [AWS IAM: List MFA Devices](#)) in the workflow.

The workflow is initiated by the data table rule, [Example: AWS IAM: Delete User For Artifact](#). For more information on this workflow, reference section [Function - AWS IAM: Delete User](#)

► Inputs:

Name	Type	Required	Example	Tooltip
<code>aws_iam_mfa_serial_nums</code>	text	No	–	–

► Outputs:

```

results = {
    'version': '1.0', 'success': True, 'reason': None,
    'content': [{ 'SerialNumber':
'arn:aws:iam::834299573936:mfa/iam_test_user', 'Status': 'OK'}],
    'raw': ' [{ 'SerialNumber':
"arn:aws:iam::834299573936:mfa/iam_test_user", "Status": "OK"}]',
    'inputs': { 'aws_iam_mfa_serial_nums':
'arn:aws:iam::834299573936:mfa/iam_test_user',
                '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': 790, 'timestamp': '2019-11-29
12:18:30'
    }
}
```

► Example Pre-Process Script:

```
content = workflow.properties.list_mfa_devices_results.content
mfa_serial_nums = []
for mfa_ser_num in content:
    if mfa_ser_num["SerialNumber"] is not None and
mfa_ser_num.get("is_virtual", None):
        mfa_serial_nums.append(mfa_ser_num["SerialNumber"])
inputs.aws_iam_mfa_serial_nums = ",".join(mfa_serial_nums)
```

► Example Post-Process Script:

```
## AWS IAM - fn_aws_iam_delete_mfa_devices script ##
# Example result:

# Globals
# List of fields in datatable for fn_aws_iam_delete_mfa_devices script
DATA_TBL_FIELDS = ["Policies"]
FN_NAME = "fn_aws_iam_delete_mfa_devices"
WF_NAME = "Delete User"
# Processing
CONTENT = results.content
INPUTS = results.inputs
QUERY_EXECUTION_DATE = results["metrics"]["timestamp"]
DEBUG_SCRIPT = False

def main():
    note_text = ''
    deleted = 0
    no_such_entity = 0
    deleted_mfas = []
    no_such_entity_mfas = []
    if CONTENT:
        for grp_stat in CONTENT:
            if grp_stat["Status"] == "OK":
                deleted += 1
                deleted_mfas.append(grp_stat["GroupName"])
            else:
                no_such_entity += 1
                no_such_entity_mfas.append(grp_stat["GroupName"])
        if deleted_mfas:
            note_text = "AWS IAM Integration: Workflow <b>{0}</b>: There
were <b>{1}</b> 'MFA devices' <b>{2}</b> deleted" \
```

```

                                "for user <b>{3}</b> for Resilient function <b>{4}
</b>." \
                                .format(WF_NAME, len(deleted_mfas), ", ".join(str(i) for i
in deleted_mfas), 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> 'MFA devices' <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_mfas), ",
".join(str(i) for i in no_such_entity_mfas), 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)
                                if DEBUG_SCRIPT:
                                    incident.addNote(helper.createRichText(note_text))
if __name__ == "__main__":
    main()

```

Function - AWS IAM: Update Login Profile

Changes 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_10

Print

Export

	Login Profile exists	Access key Ids	Groups	Policies	Tags	User Arn	Create date	Default user	Status	
_10	Yes	AKIA4EQBBG2YC3T4MROM	—	AWSDenyAll	—	arn:aws:iam::834299573936:user/iam_test_user_10	2020-03-05 17:17:00	—	Active	...

Displaying 1 - 1 of 1 (filtered from 14 total entries)

AWS IAM Access Keys

Query execution date	Access key Id	User name	Create date	Status	Default key
2020-03-11 17:01:04	AKIA4EQBBG2Y03VWDSN6	adminuser	2019-10-31 16:23:08	Active	—
2020-03-11 17:01:04	AKIA4EQBBG2YP4S4SOEV	adminuser	2019-11-04 11:33:33	Active	Yes

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: Get Access Keys

Example: AWS IAM: Refresh User

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

Cancel

Execute

Note: The minimum requirements 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 of the workflow, a note is created indicting the status of the action.

► Inputs:

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 = {'version': '1.0', 'success': True, 'reason': None,
          'content': 'OK',
          'raw': '"OK"',
          'inputs': {'aws_iam_user_name': 'iam_test_user_1',
```

```

'aws_iam_password': '***',
                    'aws_iam_password_reset_required': False},
    'metrics': {'version': '1.0', 'package': 'fn-aws-iam',
'package_version': '1.0.0',
                    'host': 'myhost.ibm.com',
                    'execution_time_ms': 4300, 'timestamp': '2020-03-13
17:35:48'
                }
}

```

► Example Pre-Process Script:

```

inputs.aws_iam_user_name = row.UserName
# Test password to see it complies with basic password policy.
err_msg_validation = "The new password needs be minimum 8 characters in
length and have at least 1 uppercase and 1 lowercase character."
err_msg_ascii = "The new password must contain only printable ASCII
characters."
if len(rule.properties.aws_iam_password) < 8:
    raise ValueError(err_msg_validation)
if not any(c.isupper() for c in rule.properties.aws_iam_password):
    raise ValueError(err_msg_validation)
if not any(c.islower() for c in rule.properties.aws_iam_password):
    raise ValueError(err_msg_validation)
try:
    rule.properties.aws_iam_password.decode('ascii')
except:
    raise ValueError(err_msg_ascii)
inputs.aws_iam_password = rule.properties.aws_iam_password
inputs.aws_iam_password_reset_required = False
if rule.properties.aws_iam_password_reset_required.lower() == "yes":
    inputs.aws_iam_password_reset_required = True

```

► Example Post-Process Script:

```

## AWS IAM - fn_aws_iam_delete_login_profile script ##

# Globals
# List of fields in datatable fn_aws_iam_update_login_profile script
FN_NAME = "fn_aws_iam_update_login_profile"
WF_NAME = "Change Profile Password"
# 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 password was 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>: Login
profile password got policy violation ERROR updating user <b>{1}</b> for " \
                "Resilient function <b>{2}</b>.".format(WF_NAME,
INPUTS["aws_iam_user_name"], FN_NAME)
        elif CONTENT == "ValidationError":
            note_text = "AWS IAM Integration: : Workflow <b>{0}</b>: Login
profile password got validation ERROR 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>: Login
profile password got unexpected ERROR 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: Delete Login Profile

Deletes 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 field 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

PrintExport

	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: Delete Login Profile

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

The 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?

CancelExecute

3. Press Execute to invoke the **Example: AWS IAM: Delete Login Profile** workflow, which calls the **AWS IAM: Delete Login Profile** function.

On successful completion of the workflow, the **Login Profile exists** field of the **AWS IAM Users** data table is updated to "No" for the selected user.

AWS IAM Users

iam_test_user_5

+ Row

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	—	AKIA4EQBBG2YNUSUX3GT	—	—	—	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 = {
  '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'
}
}
```

► Example Pre-Process Script:

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

► Example Post-Process Script:

```
## AWS IAM - fn_aws_iam_delete_login_profile script ##
# 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()
```

Data Table - AWS IAM Access Keys

AWS IAM Access Keys

Search...

PrintExport

Query execution date	Access key id	User name	Create date	Status	Default key	Last used date	Service name	Region	
There is no data for this table									

Showing 0 to 0 of 0 entries

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	-
User name	UserName	text	-
Query execution date	query_execution_date	text	-

Data Table - AWS IAM Users

AWS IAM Users

Search...

PrintExport

Query execution date	User name	Login Profile exists	Access key ids	Groups	Policies	Tags	User Arn	Create date
There is no data for this table								

Showing 0 to 0 of 0 entries

API Name:

aws_iam_users

Columns:

Column Name	API Access Name	Type	Tooltip
Access key ids	AccessKeyIds	text	-
User Arn	Arn	text	-

Column Name	API Access Name	Type	Tooltip
Create date	CreateDate	text	-
Default user	DefaultUser	text	-
Groups	Groups	text	-
Login Profile exists	LoginProfileExists	text	-
Policies	Policies	text	-
Status	Status	text	-
Tags	Tags	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 or Script Triggered
Example: AWS IAM: List Users	incident	wf_aws_iam_list_users
Example: AWS IAM: Delete User For Artifact	artifact	wf_aws_iam_delete_user_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 Key For Artifact	artifact	wf_aws_iam_delete_access_key_for_artifact
Example: AWS IAM: Get User For Artifact	artifact	wf_aws_iam_get_user_for_artifact

Rule Name	Object	Workflow or Script Triggered
Example: AWS IAM: Get access Key For Artifact	artifact	<code>wf_aws_iam_get_access_key_for_artifact</code>
Example: AWS IAM: Add User As Artifact	aws_iam_users	<code>scr_aws_iam_add_user_as_artifact</code>
Example: AWS IAM: Refresh User	aws_iam_users	<code>wf_aws_iam_refresh_user</code>
Example: AWS IAM: Attach User Policy	aws_iam_users	<code>wf_aws_iam_attach_user_policy</code>
Example: AWS IAM: Change Profile Password	aws_iam_users	<code>wf_aws_iam_change_profile_password</code>
Example: AWS IAM: Delete Access Keys	aws_iam_users	<code>wf_aws_iam_delete_access_keys</code>
Example: AWS IAM: Deactivate Access Key	aws_iam_access_keys	<code>wf_aws_iam_deactivate_access_key</code>
Example: AWS IAM: Delete User	aws_iam_users	<code>wf_aws_iam_delete_user</code>
Example: AWS IAM: Delete Login Profile	aws_iam_users	<code>wf_aws_iam_delete_login_profile</code>
Example: AWS IAM: Add Access Key As Artifact	aws_iam_access_keys	<code>scr_aws_iam_add_access_key_as_artifact</code>
Example: AWS IAM: List Access Keys	incident	<code>wf_aws_iam_list_access_keys</code>
Example: AWS IAM: Detach All User Policies	aws_iam_users	<code>wf_aws_iam_detach_all_user_policies</code>
Example: AWS IAM: Get Access Keys	aws_iam_users	<code>wf_aws_iam_get_access_keys</code>
Example: AWS IAM: Get User	aws_iam_access_keys	<code>wf_aws_iam_get_user</code>
Example: AWS IAM: Add User To Group	aws_iam_users	<code>wf_aws_iam_add_user_to_group</code>
Example: AWS IAM: Delete Access Key	aws_iam_access_keys	<code>wf_aws_iam_delete_access_key</code>

Rule Name	Object	Workflow or Script Triggered
Example: AWS IAM: Refresh Access Key	aws_iam_access_keys	wf_aws_iam_refresh_access_key