

# Package ‘aws.iam’

April 7, 2020

**Title** AWS IAM Client Package

**Version** 0.1.8

**Description** A simple client for the Amazon Web Services ('AWS') Identity and Access Management ('IAM') 'API' <<https://aws.amazon.com/iam/>>.

**License** GPL (>= 2)

**Imports** utils, httr, xml2, jsonlite, aws.signature (>= 0.3.4)

**URL** <https://github.com/cloudyr/aws.iam>

**BugReports** <https://github.com/cloudyr/aws.iam/issues>

**RoxygenNote** 7.1.0

**NeedsCompilation** no

**Author** Thomas J. Leeper [aut] (<<https://orcid.org/0000-0003-4097-6326>>),  
Simon Urbanek [cre, ctb]

**Maintainer** Simon Urbanek <[simon.urbanek@R-project.org](mailto:simon.urbanek@R-project.org)>

**Repository** CRAN

**Date/Publication** 2020-04-07 09:50:16 UTC

## R topics documented:

aws.iam-package . . . . .	2
add_policy . . . . .	2
change_pwd . . . . .	3
create_alias . . . . .	4
create_group . . . . .	5
create_key . . . . .	6
create_profile . . . . .	8
create_role . . . . .	9
create_user . . . . .	10
get_account . . . . .	11
get_session_token . . . . .	12
iamHTTP . . . . .	14
save_credentials . . . . .	15

<b>Index</b>	<b>17</b>
--------------	-----------

---

aws.iam-package	<i>aws.iam</i>
-----------------	----------------

---

**Description**

AWS IAM and STS Client Package

**Details**

A simple client package for the Amazon Web Services (AWS) Identity and Access Management (IAM) and Simple Token Service (STS) APIs.

**Author(s)**

Thomas J. Leeper <thosjleeper@gmail.com>

**References**

[IAM Documentation](#)

---

add_policy	<i>Manage IAM Policies</i>
------------	----------------------------

---

**Description**

Retrieve, create, update, and delete IAM Role, User, and Group Policies

**Usage**

```
add_policy(user, group, role, policy, doc, ...)
```

```
update_policy(role, doc, ...)
```

```
get_policy(policy, user, group, role, ...)
```

```
delete_policy(user, group, role, policy, ...)
```

```
list_policies(user, group, role, n, marker, ...)
```

**Arguments**

user	A character string specifying a user name or an object of class “iam_user”.
group	A character string containing a group name or an object of class “iam_group”.
role	A character string containing a role name or an object of class “iam_role”.
policy	A character string specifying the policy name.
doc	The contents of the policy document as a character string.
...	Additional arguments passed to <a href="#">iamHTTP</a> .
n	An integer specifying the number of responses to return.
marker	A character string specifying a marker (from a previous response) to use in paginating results

**Value**

add\_policy and get\_policy return objects of class “iam\_policy”. update\_policy and delete\_policy return a logical TRUE (if successful) or an error. list\_policies returns a list of IAM role objects.

---

change\_pwd

*Change Password*


---

**Description**

Change password for currently authenticated user

**Usage**

```
change_pwd(old, new, ...)
```

```
get_pwd_policy(...)
```

```
set_pwd_policy(
  allowchange,
  hardexpire,
  age,
  length,
  previous,
  requirements,
  ...
)
```

**Arguments**

old	A character string specifying the current password
new	A character string specifying the new password
...	Additional arguments passed to <code>iamHTTP</code> .
allowchange	Optionally, a logical indicating whether to allow users to change their own passwords (default is FALSE).
hardexpire	Optionally, a logical indicating whether to prevent users from changing their passwords after they expire (default is FALSE).
age	Optionally, a number of days (between 1 and 1095) specifying maximum valid age of an IAM user password.
length	Optionally, a minimum password length between 6 and 128 (default is 6).
previous	Optionally, a number specifying the number (between 1 and 24) of previous passwords that users are prevented from reusing. Default is 0.
requirements	A character vector specifying whether to require specific password features, including: “upper” (upper case character), “lower” (lower case character), “number” (a digit), and “symbol” (a symbol). Multiple can be specified.

**Value**

`get_pwd_policy` returns a list. `change_pwd` and `set_pwd_policy` return a logical TRUE (if successful).

**References**

[IAM Password Policies](#)

---

create_alias	<i>Manage IAM Account Aliases</i>
--------------	-----------------------------------

---

**Description**

Retrieve, create, update, and delete IAM Account Aliases

**Usage**

```
create_alias(alias, ...)
delete_alias(alias, ...)
list_aliases(n, marker, ...)
```

**Arguments**

alias	A character string specifying an alias, or an object of class “iam_alias”.
...	Additional arguments passed to <a href="#">iamHTTP</a> .
n	An integer specifying the number of responses to return.
marker	A character string specifying a marker (from a previous response) to use in paginating results

**Value**

create\_alias and delete\_alias return a logical TRUE (if successful). list\_aliases returns a list of objects of class “iam\_alias”.

**References**

[AWS Account Aliases](#)

---

create_group	<i>Manage IAM User Groups</i>
--------------	-------------------------------

---

**Description**

Retrieve, create, update, and delete IAM user groups

**Usage**

```
create_group(group, path, ...)
update_group(group, name, path, ...)
delete_group(group, ...)
get_group_users(group, n, marker, ...)
list_groups(user, n, marker, path, ...)
add_user(user, group, ...)
remove_user(user, group, ...)
```

**Arguments**

group	A character string containing a group name or an object of class “iam_group”.
path	A character string specifying a path prefix in which to locate user(s), role(s), etc. See <a href="#">Reference Identifiers</a> on the AWS Documentation for more information.
...	Additional arguments passed to <a href="#">iamHTTP</a> .

name	A character string specifying the new name for the group.
n	An integer specifying the number of responses to return.
marker	A character string specifying a marker (from a previous response) to use in paginating results
user	A character string specifying a user name.

**Value**

create\_group and get\_group return objects of class “iam\_group”. update\_group and delete\_group, add\_user, and remove\_user return a logical TRUE (if successful) or an error. list\_groups returns a list of IAM group objects. get\_group\_users returns a list of objects of class “iam\_user”, with a “iam\_group” attribute.

**See Also**

[create\\_user](#), [create\\_role](#),

**Examples**

```
## Not run:
  list_groups()

# create group
(g <- create_group("example"))
# rename
update_group(g, "example2")
list_groups()

# create example user
u <- create_user("example-user")
# add user to group
add_user(u, "example2")

get_group_users("example2")

# cleanup
remove_user(u, "example2")
delete_user(u)
delete_group("example2")

## End(Not run)
```

---

create\_key

*Manage Access Keys/Credentials*

---

**Description**

Retrieve, create, update, and delete IAM access keys

**Usage**

```
create_key(user, ...)  
  
update_key(key, user, status, ...)  
  
delete_key(key, user, ...)  
  
list_keys(user, n, marker, ...)
```

**Arguments**

user	Optionally, a character string specifying a user name or an object of class “iam_user”. This will be retrieved by default from the “UserName” list entry in key, if available; otherwise the user is assumed to be the user whose credentials are being used to execute the request.
...	Additional arguments passed to <a href="#">iamHTTP</a> .
key	A character string specifying an access key or an object of class “iam_key”.
status	A character string specifying either “Active” or “Inactive” to status the key status to.
n	An integer specifying the number of responses to return.
marker	A character string specifying a marker (from a previous response) to use in paginating results

**Value**

create\_user and get\_user return objects of class “iam\_user”. update\_user and delete\_user return a logical TRUE (if successful) or an error. list\_users returns a list of IAM user objects.

**See Also**

[create\\_user](#)

**Examples**

```
## Not run:  
# list access keys  
list_keys()  
  
# create a user key  
u <- create_user("example-user")  
str(k <- create_key(u))  
  
# toggle key status to inactive  
update_key(k, u, "Inactive")  
list_keys(u)  
  
# cleanup  
delete_key(k)  
delete_user(u)
```

```
## End(Not run)
```

---

```
create_profile      Instance Profiles
```

---

### Description

Create, retrieve, list, and delete EC2 Instance Profiles

### Usage

```
create_profile(profile, path, ...)
delete_profile(profile, ...)
get_profile(profile, ...)
list_profiles(role, n, marker, path, ...)
```

### Arguments

profile	A character string specifying the name for the profile, or an object of class “iam_instance_profile”.
path	A character string specifying a path prefix in which to locate user(s), role(s), etc. See <a href="#">Reference Identifiers</a> on the AWS Documentation for more information.
...	Additional arguments passed to <a href="#">iamHTTP</a> .
role	A character string containing a role name or an object of class “iam_role”.
n	An integer specifying the number of responses to return.
marker	A character string specifying a marker (from a previous response) to use in paginating results

### Value

An object of class “iam\_instance\_profile”.

### References

[About Instance Profiles API Documentation: CreateInstanceProfile API Documentation: DeleteInstanceProfile API Documentation: GetInstanceProfile API Documentation: ListInstanceProfiles](#)



---

create_role	<i>Manage IAM Roles</i>
-------------	-------------------------

---

### Description

Retrieve, create, update, and delete IAM Roles

### Usage

```
create_role(role, policy, path, ...)
delete_role(role, ...)
add_profile_role(role, profile, ...)
remove_profile_role(role, profile, ...)
list_roles(n, marker, path, ...)
```

### Arguments

role	A character string containing a role name or an object of class “iam_role”.
policy	...
path	A character string specifying a path prefix in which to locate user(s), role(s), etc. See <a href="#">Reference Identifiers</a> on the AWS Documentation for more information.
...	Additional arguments passed to <a href="#">iamHTTP</a> .
profile	A character string specifying the name for the profile, or an object of class “iam_instance_profile”.
n	An integer specifying the number of responses to return.
marker	A character string specifying a marker (from a previous response) to use in paginating results

### Value

create\_role and get\_role return objects of class “iam\_role”. update\_role and delete\_role return a logical TRUE (if successful) or an error. list\_roles returns a list of IAM role objects.

### See Also

[create\\_user](#), [create\\_group](#),

---

`create_user`*Manage IAM Users*

---

**Description**

Retrieve, create, update, and delete IAM Users

**Usage**

```
create_user(user, path, ...)
```

```
update_user(user, name, path, ...)
```

```
get_user(user, ...)
```

```
delete_user(user, ...)
```

```
list_users(n, marker, path, ...)
```

**Arguments**

<code>user</code>	A character string specifying a user name or an object of class “iam_user”.
<code>path</code>	A character string specifying a path prefix in which to locate user(s), role(s), etc. See <a href="#">Reference Identifiers</a> on the AWS Documentation for more information.
<code>...</code>	Additional arguments passed to <code>iamHTTP</code> .
<code>name</code>	A character string specifying the new name for the user.
<code>n</code>	An integer specifying the number of responses to return.
<code>marker</code>	A character string specifying a marker (from a previous response) to use in paginating results

**Value**

`create_user` and `get_user` return objects of class “iam\_user”. `update_user` and `delete_user` return a logical TRUE (if successful) or an error. `list_users` returns a list of IAM user objects.

**Examples**

```
## Not run:  
list_users()  
  
# create example user  
u <- create_user("example-user")  
  
# cleanup  
delete_user(u)  
  
## End(Not run)
```

---

`get_account`*Get Account Details*

---

**Description**

Retrieve IAM Account Details. This is useful as a “hello world!” test.

**Usage**

```
get_account(...)
```

```
credential_report(...)
```

```
auth_details(type, n, marker, ...)
```

**Arguments**

<code>...</code>	Additional arguments passed to <a href="#">iamHTTP</a> .
<code>type</code>	An optional character string specifying one or more types of reports to return.
<code>n</code>	An integer specifying the number of responses to return.
<code>marker</code>	A character string specifying a marker (from a previous response) to use in paginating results

**Details**

`get_account` returns a list of account details. `credential_report` generates and/or retrieves a credential report. `auth_details` returns a list of group, user, role, and policy details.

**Value**

A list containing various account details.

**Examples**

```
## Not run:  
# account details  
get_aaccount()  
  
# big list of authorizations  
auth_details()  
  
## End(Not run)
```

---

get\_session\_token      *Temporary Session Tokens*

---

### Description

Get a temporary credentials (i.e., a Session Token)

### Usage

```
get_session_token(duration = 900, id, code, tags, use = FALSE, ...)

get_federation_token(duration = 900, name, policy, use = FALSE, ...)

get_caller_identity(...)

assume_role(
  role,
  session,
  duration,
  id,
  code,
  externalid,
  policy,
  tags,
  transitive.tags,
  use = FALSE,
  ...
)
```

### Arguments

duration	numeric, optional, duration for which the credentials should be valid, in seconds, between 900 and 129600. If not set, the back-end can decide.
id	string, optional, the serial number or Amazon Resource Number for a multi-factor authentication (MFA) device.
code	If id is specified, the value provided by the MFA device.
tags	named character vector or named list of scalars, optional, if specified then the supplied key/value pairs (names are keys) are passed as session tags.
use	logical (default FALSE), specifying whether to use these credentials for subsequent requests. If TRUE, any currently used credentials are stored in a package environment (see <a href="#">save_credentials</a> ) and the requested tokens overwrite them in the relevant environment variables. <a href="#">restore_credentials</a> () can then be used to restore environment variables based on those from the saved environment and <a href="#">delete_saved_credentials</a> () deletes the credentials without restoring them.

...	Additional arguments passed to <code>stsHTTP</code> .
<code>name</code>	The name of the federated user.
<code>policy</code>	A character string specifying a JSON-formatted role policy. For <code>assume_role</code> , if <code>role</code> is an object of class “iam_role”, this will be inferred automatically.
<code>role</code>	string, role ARN or an object of class “iam_role”.
<code>session</code>	string, name of the temporary session, can be arbitrary and is mainly used to disambiguate multiple sessions using the same role.
<code>externalid</code>	A unique identifier that is used by third parties when assuming roles in their customers’ accounts.
<code>transitive.tags</code>	character vector, optional, specifies names of the session tags which will be passed to subsequent sessions in the role chain.

## Details

`get_caller_identity` returns the account ID and ARN for the currently credentialed user. This can be used to confirm that an assumed role has indeed been assumed.

`get_session_token` and `get_federation_token` generate and return temporary credentials.

Details about the underlying behavior of the various API endpoints can be found at [Requesting Temporary Security Credentials](#).

## Value

A list.

## References

[API Reference: GetCallerIdentity](#) [API Reference: GetSessionToken](#) [API Reference: GetFederationToken](#) [API Reference: AssumeRole](#) [API Reference: AssumeRoleWithSAML](#) [API Reference: AssumeRoleWithWebIdentity](#)

## Examples

```
## Not run:
get_caller_identity() # check current identity

x <- get_session_token() # get token (T1) but do not use
set_credentials(x)      # now use those credentials

x <- get_session_token(use = TRUE) # get and use another temp token (T2)
get_caller_identity() # check that token is in use

# assume a role
r <- assume_role("arn:aws:iam::111111111111:role/my-role", "test", use=TRUE)
get_caller_identity() # check that the role has been assumed

restore_credentials() # return to credentials of T2
restore_credentials() # return to credentials of T1
```

```

restore_credentials() # return to root credentials
get_caller_identity() # check identity, again

## End(Not run)

```

---

iamHTTP

*Workhorse API Query Functions*


---

## Description

These are the low-level API querying functions for IAM and STS. Users do not need to use these directly.

## Usage

```

iamHTTP(
  verb = "GET",
  query,
  headers = list(),
  body = "",
  version = "2010-05-08",
  verbose = getOption("verbose", FALSE),
  region = Sys.getenv("AWS_DEFAULT_REGION", "us-east-1"),
  key = NULL,
  secret = NULL,
  session_token = NULL,
  ...
)

stsHTTP(
  query,
  headers = list(),
  body = "",
  version = "2011-06-15",
  verbose = getOption("verbose", FALSE),
  region = Sys.getenv("AWS_DEFAULT_REGION", "us-east-1"),
  key = NULL,
  secret = NULL,
  session_token = NULL,
  ...
)

```

## Arguments

verb	A character string specifying an HTTP verb. Either "GET" or "POST".
query	A named list specifying query arguments.
headers	A list of headers to pass to the HTTP request.

body	A character string specifying a request body (if verb = "POST").
version	A character string specifying an API version. Default is "2010-05-08".
verbose	A logical indicating whether to be verbose. Default is given by options("verbose").
region	A character string specifying an AWS region. See <a href="#">locate_credentials</a> .
key	A character string specifying an AWS Access Key. See <a href="#">locate_credentials</a> .
secret	A character string specifying an AWS Secret Key. See <a href="#">locate_credentials</a> .
session_token	Optionally, a character string specifying an AWS temporary Session Token to use in signing a request. See <a href="#">locate_credentials</a> .
...	Additional arguments passed to <a href="#">GET</a> or <a href="#">POST</a>

---

save_credentials	<i>Save/restore/manage session credentials</i>
------------------	--

---

## Description

The following functions manage the environment variables `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and `AWS_SESSION_TOKEN` used for credentials for all AWS API calls.

`save_credentials` saves the current credentials to a stack of credentials kept in the session. Always returns TRUE.

`restore_credentials` restores the last saved credentials and pops them off the stack.

`delete_saved_credentials` removes the last saved credentials without using them.

`set_credentials` uses credentials list as supplied by the REST API and makes them current by assigning their values to the corresponding `AWS_*` environment variables. If `save.previous` is TRUE then the currently used credentials are first saved on the stack before being replaced with the new ones.

Most functions in the STS section call `set_credentials()` automatically if `use = TRUE` is set.

## Usage

```
save_credentials()
```

```
set_credentials(credentials, save.previous = TRUE)
```

```
delete_saved_credentials(all = FALSE)
```

```
restore_credentials(pop = TRUE, root = FALSE)
```

## Arguments

`credentials` list, credentials as received from the REST API call, they should contain to following elements: `AccessKeyId`, `SecretAccessKey` and `SessionToken`)

`save.previous` logical, if TRUE the current credentials are saved first using `save_credentials` before the new credentials are applied.

<code>all</code>	logical, if TRUE then removes all credentials from the stack, otherwise only the last ones.
<code>pop</code>	logical, if TRUE then the credentials are restored and then removed from the stack.
<code>root</code>	logical, if FALSE then last saved credentials are used. if TRUE then goes down the stack to the first saved credentials. If both <code>root</code> and <code>pop</code> are TRUE then all credentials are removed from the stack.

**Details**

Since `aws.iam` version 0.1.8 the credentials are kept on a stack, so it is possible to use `save_credentials()` several times without restoring them. This allows role chaining. At the end of a chained session it is possible to get back to the main credentials using `restore_credentials(pop=TRUE, root=TRUE)`.



# Index

## \*Topic **package**

- aws.iam-package, 2
- add\_policy, 2
- add\_profile\_role (create\_role), 9
- add\_user (create\_group), 5
- assume\_role (get\_session\_token), 12
- auth\_details (get\_account), 11
- aws.iam (aws.iam-package), 2
- aws.iam-package, 2
- change\_pwd, 3
- create\_alias, 4
- create\_group, 5, 9
- create\_key, 6
- create\_profile, 8
- create\_role, 6, 9
- create\_user, 6, 7, 9, 10
- credential\_report (get\_account), 11
- delete\_alias (create\_alias), 4
- delete\_group (create\_group), 5
- delete\_key (create\_key), 6
- delete\_policy (add\_policy), 2
- delete\_profile (create\_profile), 8
- delete\_role (create\_role), 9
- delete\_saved\_credentials, 12
- delete\_saved\_credentials (save\_credentials), 15
- delete\_user (create\_user), 10
- GET, 15
- get\_account, 11
- get\_caller\_identity (get\_session\_token), 12
- get\_federation\_token (get\_session\_token), 12
- get\_group\_users (create\_group), 5
- get\_policy (add\_policy), 2
- get\_profile (create\_profile), 8
- get\_pwd\_policy (change\_pwd), 3
- get\_session\_token, 12
- get\_user (create\_user), 10
- iamHTTP, 3–5, 7–11, 14
- list\_aliases (create\_alias), 4
- list\_groups (create\_group), 5
- list\_keys (create\_key), 6
- list\_policies (add\_policy), 2
- list\_profiles (create\_profile), 8
- list\_roles (create\_role), 9
- list\_users (create\_user), 10
- locate\_credentials, 15
- POST, 15
- remove\_profile\_role (create\_role), 9
- remove\_user (create\_group), 5
- restore\_credentials, 12
- restore\_credentials (save\_credentials), 15
- save\_credentials, 12, 15
- set\_credentials (save\_credentials), 15
- set\_pwd\_policy (change\_pwd), 3
- stsHTTP, 13
- stsHTTP (iamHTTP), 14
- update\_group (create\_group), 5
- update\_key (create\_key), 6
- update\_policy (add\_policy), 2
- update\_user (create\_user), 10