
IAM Floyd

Release 0.170.0

Daniel Schroeder

Apr 27, 2021

CONTENTS

1	Getting Started	3
2	Vocabulary	17
2.1	allow deny (Effect)	17
2.2	to (Action)	18
2.3	all (Action)	19
2.3.1	allActions	19
2.3.2	allMatchingActions	19
2.3.3	Access levels	21
2.4	if (Condition)	31
2.4.1	Operators	33
2.5	on (Resource)	35
2.6	for (Principal)	36
2.7	not (notAction, notResource and notPrincipal)	42
2.7.1	notActions	42
2.7.2	notResources	42
2.7.3	notPrincipals	43
2.8	compact	44
3	Examples	45
4	Collections	53
4.1	Available collections	54
4.1.1	allowEc2InstanceDeleteByOwner	54
5	Frequently Asked Questions	55
5.1	Why should I use this package instead of writing policies by hand?	55
5.2	Are all actions / conditions / resource types covered?	56
5.3	How often will there be updates to reflect IAM changes?	57
5.4	Do you release new packages when a new CDK version is released?	57
5.5	Is the package following semantic versioning?	57
5.6	I don't like method chaining!	57
5.7	Floyd?	58
6	Legal	59
6.1	License	59
7	IAM Floyd	61
7.1	Similar projects	61

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

There are two different package variants available:

iam-floyd:

Can be used in AWS SDK, Boto 3 or for whatever you need an IAM policy statement for:

cdk-iam-floyd:

Integrates into [AWS CDK](#) and extends `iam.PolicyStatement`:

Find them all on [libraries.io](#).

Java packages currently are only available on [GitHub](#).

GETTING STARTED

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

Note: Use the online [policy converter](#) to migrate any JSON policy to Floyd code!

Depending on your scenario, you need to either install/import `iam-floyd` or `cdk-iam-floyd`:

JavaScript

Python

```
# for use without AWS CDK use the iam-floyd package
npm install iam-floyd

# for use with CDK use the cdk-iam-floyd package
npm install cdk-iam-floyd
```

```
# for use without AWS CDK use the iam-floyd package
pip install iam-floyd

# for use with CDK use the cdk-iam-floyd package
pip install cdk-iam-floyd
```

JavaScript

TypeScript

Python

```
// for use without AWS CDK use the iam-floyd package
var statement = require('iam-floyd');

// for use with CDK use the cdk-iam-floyd package
var statement = require('cdk-iam-floyd');
```

```
// for use without AWS CDK use the iam-floyd package
import * as statement from 'iam-floyd';

// for use with CDK use the cdk-iam-floyd package
import * as statement from 'cdk-iam-floyd';
```

```
# for use without AWS CDK use the iam-floyd package
import iam_floyd as statement

# for use with CDK use the cdk-iam-floyd package
import cdk_iam_floyd as statement
```

Both packages contain a statement provider for each AWS service, e.g. `Ec2`. A statement provider is a class with methods for each and every available action, resource type and condition. Calling such method will add the action/resource/condition to the statement:

JavaScript

Python

Result

```
new statement Ec2().toStartInstances()
```

```
statement.Ec2().to_start_instances()
```

```
{
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}
```

Every method returns the statement provider, so you can chain method calls:

JavaScript

Python

Result

```
new statement Ec2() //
  toStartInstances()
  toStopInstances()
```

```
statement.Ec2() \
  .to_start_instances() \
  .to_stop_instances()
```

```
{
  "Action": [
    "ec2:StartInstances",
    "ec2:StopInstances"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

The default effect of any statement is `Allow`. To add some linguistic sugar you can explicitly call the `allow()` method:

JavaScript

Python

Result

```
new statement Ec2 () //
  allow()
  toStartInstances()
  toStopInstances()
```

```
statement.Ec2 () \
  .allow() \
  .to_start_instances() \
  .to_stop_instances()
```

```
{
  "Action": [
    "ec2:StartInstances",
    "ec2:StopInstances"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

Or deny ():

JavaScript

Python

Result

```
new statement Ec2 () //
  deny()
  toStartInstances()
  toStopInstances()
```

```
statement.Ec2 () \
  .deny() \
  .to_start_instances() \
  .to_stop_instances()
```

```
{
  "Action": [
    "ec2:StartInstances",
    "ec2:StopInstances"
  ],
  "Resource": "*",
  "Effect": "Deny"
}
```

You can work with [access levels](#). For every access level there are distinct methods available to add all related actions to the statement:

JavaScript

Python

- allListActions()
- allReadActions()
- allWriteActions()
- allPermissionManagementActions()

- `allTaggingActions()`
- `all_list_actions()`
- `all_read_actions()`
- `all_write_actions()`
- `all_permission_management_actions()`
- `all_tagging_actions()`

JavaScript

Python

Result

```
const s1 = new statement.Ec2() //
    .deny()
    .allPermissionManagementActions();

const s2 = new statement.Ec2() //
    .allow()
    .allListActions()
    .allReadActions();
```

```
s1 = statement.Ec2() \
    .deny() \
    .all_permission_management_actions()

s2 = statement.Ec2() \
    .allow() \
    .all_list_actions() \
    .all_read_actions()
```

```
{
  "Action": [
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterfacePermission",
    "ec2:ModifySnapshotAttribute",
    "ec2:ModifyVpcEndpointServicePermissions",
    "ec2:ResetSnapshotAttribute"
  ],
  "Resource": "*",
  "Effect": "Deny"
}

{
  "Action": [
    "ec2:DescribeAccountAttributes",
    "ec2:DescribeAddresses",
    "ec2:DescribeAddressesAttribute",
    "ec2:DescribeAggregateIdFormat",
    "ec2:DescribeAvailabilityZones",
    "ec2:DescribeBundleTasks",
    "ec2:DescribeByoipCidrs",
    "ec2:DescribeCapacityReservations",
    "ec2:DescribeCarrierGateways",
    "ec2:DescribeClassicLinkInstances",
    "ec2:DescribeClientVpnAuthorizationRules",
```

(continues on next page)

(continued from previous page)

```
"ec2:DescribeClientVpnConnections",
"ec2:DescribeClientVpnEndpoints",
"ec2:DescribeClientVpnRoutes",
"ec2:DescribeClientVpnTargetNetworks",
"ec2:DescribeCoipPools",
"ec2:DescribeConversionTasks",
"ec2:DescribeCustomerGateways",
"ec2:DescribeDhcpOptions",
"ec2:DescribeEgressOnlyInternetGateways",
"ec2:DescribeElasticGpus",
"ec2:DescribeExportImageTasks",
"ec2:DescribeExportTasks",
"ec2:DescribeFastSnapshotRestores",
"ec2:DescribeFleetHistory",
"ec2:DescribeFleetInstances",
"ec2:DescribeFleets",
"ec2:DescribeFlowLogs",
"ec2:DescribeFpgaImageAttribute",
"ec2:DescribeFpgaImages",
"ec2:DescribeHostReservationOfferings",
"ec2:DescribeHostReservations",
"ec2:DescribeHosts",
"ec2:DescribeIamInstanceProfileAssociations",
"ec2:DescribeIdFormat",
"ec2:DescribeIdentityIdFormat",
"ec2:DescribeImageAttribute",
"ec2:DescribeImages",
"ec2:DescribeImportImageTasks",
"ec2:DescribeImportSnapshotTasks",
"ec2:DescribeInstanceAttribute",
"ec2:DescribeInstanceCreditSpecifications",
"ec2:DescribeInstanceEventNotificationAttributes",
"ec2:DescribeInstanceStatus",
"ec2:DescribeInstanceTypeOfferings",
"ec2:DescribeInstanceTypes",
"ec2:DescribeInstances",
"ec2:DescribeInternetGateways",
"ec2:DescribeIpv6Pools",
"ec2:DescribeKeyPairs",
"ec2:DescribeLaunchTemplateVersions",
"ec2:DescribeLaunchTemplates",
"ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations",
"ec2:DescribeLocalGatewayRouteTableVpcAssociations",
"ec2:DescribeLocalGatewayRouteTables",
"ec2:DescribeLocalGatewayVirtualInterfaceGroups",
"ec2:DescribeLocalGatewayVirtualInterfaces",
"ec2:DescribeLocalGateways",
"ec2:DescribeManagedPrefixLists",
"ec2:DescribeMovingAddresses",
"ec2:DescribeNatGateways",
"ec2:DescribeNetworkAcls",
"ec2:DescribeNetworkInsightsAnalyses",
"ec2:DescribeNetworkInsightsPaths",
"ec2:DescribeNetworkInterfaceAttribute",
"ec2:DescribeNetworkInterfacePermissions",
"ec2:DescribeNetworkInterfaces",
"ec2:DescribePlacementGroups",
```

(continues on next page)

(continued from previous page)

```
"ec2:DescribePrefixLists",
"ec2:DescribePrincipalIdFormat",
"ec2:DescribePublicIpv4Pools",
"ec2:DescribeRegions",
"ec2:DescribeReservedInstances",
"ec2:DescribeReservedInstancesListings",
"ec2:DescribeReservedInstancesModifications",
"ec2:DescribeReservedInstancesOfferings",
"ec2:DescribeRouteTables",
"ec2:DescribeScheduledInstanceAvailability",
"ec2:DescribeScheduledInstances",
"ec2:DescribeSecurityGroupReferences",
"ec2:DescribeSecurityGroups",
"ec2:DescribeSnapshotAttribute",
"ec2:DescribeSnapshots",
"ec2:DescribeSpotDatafeedSubscription",
"ec2:DescribeSpotFleetInstances",
"ec2:DescribeSpotFleetRequestHistory",
"ec2:DescribeSpotFleetRequests",
"ec2:DescribeSpotInstanceRequests",
"ec2:DescribeSpotPriceHistory",
"ec2:DescribeStaleSecurityGroups",
"ec2:DescribeSubnets",
"ec2:DescribeTags",
"ec2:DescribeTrafficMirrorFilters",
"ec2:DescribeTrafficMirrorSessions",
"ec2:DescribeTrafficMirrorTargets",
"ec2:DescribeTransitGatewayAttachments",
"ec2:DescribeTransitGatewayConnectPeers",
"ec2:DescribeTransitGatewayConnects",
"ec2:DescribeTransitGatewayMulticastDomains",
"ec2:DescribeTransitGatewayPeeringAttachments",
"ec2:DescribeTransitGatewayRouteTables",
"ec2:DescribeTransitGatewayVpcAttachments",
"ec2:DescribeTransitGateways",
"ec2:DescribeVolumeAttribute",
"ec2:DescribeVolumeStatus",
"ec2:DescribeVolumes",
"ec2:DescribeVolumesModifications",
"ec2:DescribeVpcAttribute",
"ec2:DescribeVpcClassicLink",
"ec2:DescribeVpcClassicLinkDnsSupport",
"ec2:DescribeVpcEndpointConnectionNotifications",
"ec2:DescribeVpcEndpointConnections",
"ec2:DescribeVpcEndpointServiceConfigurations",
"ec2:DescribeVpcEndpointServicePermissions",
"ec2:DescribeVpcEndpointServices",
"ec2:DescribeVpcEndpoints",
"ec2:DescribeVpcPeeringConnections",
"ec2:DescribeVpcs",
"ec2:DescribeVpnConnections",
"ec2:DescribeVpnGateways",
"ec2:ExportClientVpnClientCertificateRevocationList",
"ec2:ExportClientVpnClientConfiguration",
"ec2:GetAssociatedEnclaveCertificateIamRoles",
"ec2:GetAssociatedIpv6PoolCidrs",
"ec2:GetCapacityReservationUsage"
```

(continues on next page)

(continued from previous page)

```

        "ec2:GetCoipPoolUsage",
        "ec2:GetConsoleOutput",
        "ec2:GetConsoleScreenshot",
        "ec2:GetDefaultCreditSpecification",
        "ec2:GetEbsDefaultKmsKeyId",
        "ec2:GetEbsEncryptionByDefault",
        "ec2:GetGroupsForCapacityReservation",
        "ec2:GetHostReservationPurchasePreview",
        "ec2:GetLaunchTemplateData",
        "ec2:GetManagedPrefixListAssociations",
        "ec2:GetManagedPrefixListEntries",
        "ec2:GetPasswordData",
        "ec2:GetReservedInstancesExchangeQuote",
        "ec2:GetSerialConsoleAccessStatus",
        "ec2:GetTransitGatewayAttachmentPropagations",
        "ec2:GetTransitGatewayMulticastDomainAssociations",
        "ec2:GetTransitGatewayPrefixListReferences",
        "ec2:GetTransitGatewayRouteTableAssociations",
        "ec2:GetTransitGatewayRouteTablePropagations",
        "ec2:SearchLocalGatewayRoutes",
        "ec2:SearchTransitGatewayMulticastGroups",
        "ec2:SearchTransitGatewayRoutes"
    ],
    "Resource": "*",
    "Effect": "Allow"
}

```

To add actions based on regular expressions, use the method `allMatchingActions()`.

Important: No matter in which language you use the package, the regular expressions need to be in [Perl/JavaScript literal style](#) and need to be passed as strings!

JavaScript

Python

Result

```

new statement Ec2() //
  deny()
  allMatchingActions('/vpn/i')

```

```

statement.Ec2() \
  .deny() \
  .all_matching_actions('/vpn/i')

```

```

{
  "Action": [
    "ec2:ApplySecurityGroupsToClientVpnTargetNetwork",
    "ec2:AssociateClientVpnTargetNetwork",
    "ec2:AttachVpnGateway",
    "ec2:AuthorizeClientVpnIngress",
    "ec2:CreateClientVpnEndpoint",
    "ec2:CreateClientVpnRoute",
    "ec2:CreateVpnConnection",

```

(continues on next page)

(continued from previous page)

```

        "ec2:CreateVpnConnectionRoute",
        "ec2:CreateVpnGateway",
        "ec2>DeleteClientVpnEndpoint",
        "ec2>DeleteClientVpnRoute",
        "ec2>DeleteVpnConnection",
        "ec2>DeleteVpnConnectionRoute",
        "ec2>DeleteVpnGateway",
        "ec2:DescribeClientVpnAuthorizationRules",
        "ec2:DescribeClientVpnConnections",
        "ec2:DescribeClientVpnEndpoints",
        "ec2:DescribeClientVpnRoutes",
        "ec2:DescribeClientVpnTargetNetworks",
        "ec2:DescribeVpnConnections",
        "ec2:DescribeVpnGateways",
        "ec2:DetachVpnGateway",
        "ec2:DisassociateClientVpnTargetNetwork",
        "ec2:ExportClientVpnClientCertificateRevocationList",
        "ec2:ExportClientVpnClientConfiguration",
        "ec2:ImportClientVpnClientCertificateRevocationList",
        "ec2:ModifyClientVpnEndpoint",
        "ec2:ModifyVpnConnection",
        "ec2:ModifyVpnConnectionOptions",
        "ec2:ModifyVpnTunnelCertificate",
        "ec2:ModifyVpnTunnelOptions",
        "ec2:RevokeClientVpnIngress",
        "ec2:TerminateClientVpnConnections"
    ],
    "Resource": "*",
    "Effect": "Deny"
}

```

To add all actions (e.g. `ec2:*`), call the `allActions()` method:

JavaScript

Python

Result

```

new statement Ec2() //
    allow()
    allActions()

```

```

statement.Ec2() \
    .allow() \
    .all_actions()

```

```

{
    "Action": "ec2:*",
    "Resource": "*",
    "Effect": "Allow"
}

```

For every available condition key, there are `if*`() methods available.

JavaScript

Python

Result

```
new statement Ec2()
  allow()
  toStartInstances()
  ifEncrypted()
  ifInstanceType(['t3.micro', 't3.nano'])
  ifAssociatePublicIpAddress(false)
  ifAwsRequestTag('Owner', 'John')
```

```
statement.Ec2() \
  .allow() \
  .to_start_instances() \
  .if_encrypted() \
  .if_instance_type(['t3.micro', 't3.nano']) \
  .if_associate_public_ip_address(False) \
  .if_aws_request_tag('Owner', 'John')
```

```
{
  "Condition": {
    "Bool": {
      "ec2:Encrypted": "true",
      "ec2:AssociatePublicIpAddress": "false"
    },
    "StringLike": {
      "ec2:InstanceType": [
        "t3.micro",
        "t3.nano"
      ],
      "aws:RequestTag/Owner": "John"
    }
  },
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}
```

To add a condition not covered by the available methods, you can define just any condition yourself via `if()`:

JavaScript

Python

Result

```
new statement Ec2()
  allow()
  toStartInstances()
  if('ec2:missingCondition', 'some-value')
```

```
statement.Ec2() \
  .allow() \
  .to_start_instances() \
  .if_('ec2:missingCondition', 'some-value')
```

```
{
  "Condition": {
```

(continues on next page)

(continued from previous page)

```

    "StringLike": {
      "ec2:missingCondition": "some-value"
    },
    "Action": "ec2:StartInstances",
    "Resource": "*",
    "Effect": "Allow"
  }

```

The default operator for conditions of type `String` is `StringLike`.

Most of the `if*`() methods allow an optional operator as last argument:

JavaScript

Python

Result

```

new statement Ec2()
  allow()
  toStartInstances()
  ifAwsRequestTag('TagWithSpecialChars', '*John*', 'StringEquals')

```

```

statement.Ec2() \
  .allow() \
  .to_start_instances() \
  .if_aws_request_tag('TagWithSpecialChars', '*John*', 'StringEquals')

```

```

{
  "Condition": {
    "StringEquals": {
      "aws:RequestTag/TagWithSpecialChars": "*John*"
    }
  },
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}

```

Statements without principals, by default, apply to all resources. To limit to specific resources, add them via `on*`(). For every resource type an `on*`() method exists:

JavaScript

Python

Result

```

new statement S3()
  allow()
  allActions()
  onBucket('example-bucket')
  onObject('example-bucket', 'some/path/*')

```

```

statement.S3() \
  .allow() \
  .all_actions() \

```

(continues on next page)

(continued from previous page)

```
.on_bucket('example-bucket') \
.on_object('example-bucket', 'some/path/*')
```

```
{
  "Action": "s3:*",
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/some/path/*"
  ],
  "Effect": "Allow"
}
```

If instead you have an ARN ready, use the `on()` method:

JavaScript

Python

Result

```
new statement S3() //
  allow()
  allActions()
  on(
    'arn:aws:s3:::example-bucket', //
    'arn:aws:s3:::another-bucket'
  )
```

```
statement.S3() \
  .allow() \
  .all_actions() \
  .on('arn:aws:s3:::example-bucket',
      'arn:aws:s3:::another-bucket')
```

```
{
  "Action": "s3:*",
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::another-bucket"
  ],
  "Effect": "Allow"
}
```

To invert the policy you can use `notActions()`, `notResources()` and `notPrincipals()`:

JavaScript

Python

Result

```
new statement S3()
  allow()
  notActions()
  toDeleteBucket()
  onBucket('example-bucket')
```

```
statement.S3() \
    .allow() \
    .not_actions() \
    .to_delete_bucket() \
    .on_bucket('example-bucket')
```

```
{
  "NotAction": "s3:DeleteBucket",
  "Resource": "arn:aws:s3:::example-bucket",
  "Effect": "Allow"
}
```

JavaScript

Python

Result

```
new statement.S3()
  .allow()
  .notResources()
  .toDeleteBucket()
  .onBucket('example-bucket')
```

```
statement.S3() \
    .allow() \
    .not_resources() \
    .to_delete_bucket() \
    .on_bucket('example-bucket')
```

```
{
  "Action": "s3:DeleteBucket",
  "NotResource": "arn:aws:s3:::example-bucket",
  "Effect": "Allow"
}
```

JavaScript

Python

Result

```
new statement.S3()
  .allow()
  .allActions()
  .notPrincipals()
  .forUser('1234567890', 'Bob')
  .onObject('example-bucket', '*')
```

```
statement.S3() \
    .allow() \
    .all_actions() \
    .not_principals() \
    .for_user('1234567890', 'Bob')
    .on_object('example-bucket', '*')
```

```
{  
  "Action": "s3:*",  
  "Resource": "arn:aws:s3:::example-bucket/*",  
  "Effect": "Allow",  
  "NotPrincipal": {  
    "AWS": [  
      "arn:aws:iam::1234567890:user/Bob"  
    ]  
  }  
}
```


VOCABULARY

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

IAM Floyd provides a fluid interface and enables you to define policy statements in a human readable and easy to understand phrase.

2.1 allow | deny (Effect)

The methods `allow()` and `deny()` control the **Effect** of the statement.

The default effect of any statement is `Allow`, so it's not mandatory to add either of these methods to the method chain. Though it is recommended to improve readability:

JavaScript

Python

Result

```
const s1 = new statement.Ec2() //
    .allow()
    .toStartInstances();

const s2 = new statement.Ec2() //
    .deny()
    .toStopInstances();
```

```
s1 = statement.Ec2() \
    .allow() \
    .to_start_instances()

s2 = statement.Ec2() \
    .deny() \
    .to_stop_instances()
```

```
{
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}
```

(continues on next page)

(continued from previous page)

```

}
{
  "Action": "ec2:StopInstances",
  "Resource": "*",
  "Effect": "Deny"
}

```

2.2 to (Action)

Every available IAM **action** is represented by a distinct method. These methods start with `to`. You allow/deny **to do something**

JavaScript

Python

Result

```

new statement Ec2() //
  allow()
  toStartInstances()
  toStopInstances()

```

```

statement.Ec2() \
  .allow() \
  .to_start_instances() \
  .to_stop_instances()

```

```

{
  "Action": [
    "ec2:StartInstances",
    "ec2:StopInstances"
  ],
  "Resource": "*",
  "Effect": "Allow"
}

```

In case of **missing actions**, you can just add any action key yourself via `to()`:

JavaScript

Python

Result

```

new statement Ec2() //
  allow()
  to('missingAction')

```

```

statement.Ec2() \
  .allow() \
  .to('missingAction')

```

```
{
  "Action": "ec2:missingAction",
  "Resource": "*",
  "Effect": "Allow"
}
```

2.3 all (Action)

While methods starting with `to` add a single action to a statement, methods starting with `all` add multiple [actions](#).

2.3.1 allActions

This method adds all actions of the related service to the statement, e.g. `ec2:*`

JavaScript

Python

Result

```
new statement Ec2() //
  .allow()
  .allActions()
```

```
statement.Ec2() \
  .allow() \
  .all_actions()
```

```
{
  "Action": "ec2:*",
  "Resource": "*",
  "Effect": "Allow"
}
```

2.3.2 allMatchingActions

Adds all actions matching regular expressions to the statement.

Attention: The list of actions is compiled at run time. The generated statement object contains an exact list of actions that matched when you build it. If AWS later adds/removes actions that would match the regular expression, you need to re-generate the statements.

The regular expressions need to be in [Perl/JavaScript literal style](#) and need to be passed as strings:

JavaScript

Python

Result

```
new statement Ec2() //
  deny()
  allMatchingActions('/vpn/i')
```

```
statement.Ec2() \
  .deny() \
  .all_matching_actions('/vpn/i')
```

```
{
  "Action": [
    "ec2:ApplySecurityGroupsToClientVpnTargetNetwork",
    "ec2:AssociateClientVpnTargetNetwork",
    "ec2:AttachVpnGateway",
    "ec2:AuthorizeClientVpnIngress",
    "ec2:CreateClientVpnEndpoint",
    "ec2:CreateClientVpnRoute",
    "ec2:CreateVpnConnection",
    "ec2:CreateVpnConnectionRoute",
    "ec2:CreateVpnGateway",
    "ec2>DeleteClientVpnEndpoint",
    "ec2>DeleteClientVpnRoute",
    "ec2>DeleteVpnConnection",
    "ec2>DeleteVpnConnectionRoute",
    "ec2>DeleteVpnGateway",
    "ec2:DescribeClientVpnAuthorizationRules",
    "ec2:DescribeClientVpnConnections",
    "ec2:DescribeClientVpnEndpoints",
    "ec2:DescribeClientVpnRoutes",
    "ec2:DescribeClientVpnTargetNetworks",
    "ec2:DescribeVpnConnections",
    "ec2:DescribeVpnGateways",
    "ec2:DetachVpnGateway",
    "ec2:DisassociateClientVpnTargetNetwork",
    "ec2:ExportClientVpnClientCertificateRevocationList",
    "ec2:ExportClientVpnClientConfiguration",
    "ec2:ImportClientVpnClientCertificateRevocationList",
    "ec2:ModifyClientVpnEndpoint",
    "ec2:ModifyVpnConnection",
    "ec2:ModifyVpnConnectionOptions",
    "ec2:ModifyVpnTunnelCertificate",
    "ec2:ModifyVpnTunnelOptions",
    "ec2:RevokeClientVpnIngress",
    "ec2:TerminateClientVpnConnections"
  ],
  "Resource": "*",
  "Effect": "Deny"
}
```

2.3.3 Access levels

To add all actions of a certain [access level](#) to the statement use the below methods.

Attention: The list of actions is compiled at run time. The generated statement object contains an exact list of actions that matched when you build it. If AWS later adds/removes actions or changes the level, you need to re-generate the statements.

Note: When working with access levels the policy size limits may be exceeded quickly, just because there are so many actions available for some services like EC2.

In these cases you should use the [compact](#) method, to compile the action list to a list of wildcard patterns.

allListActions

Adds all actions with [access level](#) **list** to the statement.

JavaScript

Python

Result

```
new statement.Ec2() //
  allow()
  allListActions()
```

```
statement.Ec2() \
  .allow() \
  .all_list_actions()
```

```
{
  "Action": [
    "ec2:DescribeAccountAttributes",
    "ec2:DescribeAddresses",
    "ec2:DescribeAddressesAttribute",
    "ec2:DescribeAggregateIdFormat",
    "ec2:DescribeAvailabilityZones",
    "ec2:DescribeBundleTasks",
    "ec2:DescribeByoipCidrs",
    "ec2:DescribeCapacityReservations",
    "ec2:DescribeCarrierGateways",
    "ec2:DescribeClassicLinkInstances",
    "ec2:DescribeClientVpnAuthorizationRules",
    "ec2:DescribeClientVpnConnections",
    "ec2:DescribeClientVpnEndpoints",
    "ec2:DescribeClientVpnRoutes",
    "ec2:DescribeClientVpnTargetNetworks",
    "ec2:DescribeCoipPools",
    "ec2:DescribeConversionTasks",
    "ec2:DescribeCustomerGateways",
    "ec2:DescribeDhcpOptions",
    "ec2:DescribeEgressOnlyInternetGateways",
```

(continues on next page)

(continued from previous page)

```

"ec2:DescribeExportImageTasks",
"ec2:DescribeExportTasks",
"ec2:DescribeFleetHistory",
"ec2:DescribeFleetInstances",
"ec2:DescribeFleets",
"ec2:DescribeFlowLogs",
"ec2:DescribeFpgaImageAttribute",
"ec2:DescribeFpgaImages",
"ec2:DescribeHostReservationOfferings",
"ec2:DescribeHostReservations",
"ec2:DescribeHosts",
"ec2:DescribeIamInstanceProfileAssociations",
"ec2:DescribeIdFormat",
"ec2:DescribeIdentityIdFormat",
"ec2:DescribeImageAttribute",
"ec2:DescribeImages",
"ec2:DescribeImportImageTasks",
"ec2:DescribeImportSnapshotTasks",
"ec2:DescribeInstanceAttribute",
"ec2:DescribeInstanceCreditSpecifications",
"ec2:DescribeInstanceEventNotificationAttributes",
"ec2:DescribeInstanceState",
"ec2:DescribeInstanceTypeOfferings",
"ec2:DescribeInstanceTypes",
"ec2:DescribeInstances",
"ec2:DescribeInternetGateways",
"ec2:DescribeIpv6Pools",
"ec2:DescribeKeyPairs",
"ec2:DescribeLaunchTemplateVersions",
"ec2:DescribeLaunchTemplates",
"ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations",
"ec2:DescribeLocalGatewayRouteTableVpcAssociations",
"ec2:DescribeLocalGatewayRouteTables",
"ec2:DescribeLocalGatewayVirtualInterfaceGroups",
"ec2:DescribeLocalGatewayVirtualInterfaces",
"ec2:DescribeLocalGateways",
"ec2:DescribeManagedPrefixLists",
"ec2:DescribeMovingAddresses",
"ec2:DescribeNatGateways",
"ec2:DescribeNetworkAcls",
"ec2:DescribeNetworkInsightsAnalyses",
"ec2:DescribeNetworkInsightsPaths",
"ec2:DescribeNetworkInterfaceAttribute",
"ec2:DescribeNetworkInterfacePermissions",
"ec2:DescribeNetworkInterfaces",
"ec2:DescribePlacementGroups",
"ec2:DescribePrefixLists",
"ec2:DescribePrincipalIdFormat",
"ec2:DescribePublicIpv4Pools",
"ec2:DescribeRegions",
"ec2:DescribeReservedInstances",
"ec2:DescribeReservedInstancesListings",
"ec2:DescribeReservedInstancesModifications",
"ec2:DescribeReservedInstancesOfferings",
"ec2:DescribeRouteTables",
"ec2:DescribeSecurityGroupReferences",
"ec2:DescribeSecurityGroups",

```

(continues on next page)

(continued from previous page)

```

    "ec2:DescribeSnapshotAttribute",
    "ec2:DescribeSnapshots",
    "ec2:DescribeSpotDatafeedSubscription",
    "ec2:DescribeSpotFleetInstances",
    "ec2:DescribeSpotFleetRequestHistory",
    "ec2:DescribeSpotFleetRequests",
    "ec2:DescribeSpotInstanceRequests",
    "ec2:DescribeSpotPriceHistory",
    "ec2:DescribeStaleSecurityGroups",
    "ec2:DescribeSubnets",
    "ec2:DescribeTrafficMirrorFilters",
    "ec2:DescribeTrafficMirrorSessions",
    "ec2:DescribeTrafficMirrorTargets",
    "ec2:DescribeTransitGatewayAttachments",
    "ec2:DescribeTransitGatewayConnectPeers",
    "ec2:DescribeTransitGatewayConnects",
    "ec2:DescribeTransitGatewayMulticastDomains",
    "ec2:DescribeTransitGatewayPeeringAttachments",
    "ec2:DescribeTransitGatewayRouteTables",
    "ec2:DescribeTransitGatewayVpcAttachments",
    "ec2:DescribeTransitGateways",
    "ec2:DescribeVolumeAttribute",
    "ec2:DescribeVolumeStatus",
    "ec2:DescribeVolumes",
    "ec2:DescribeVpcAttribute",
    "ec2:DescribeVpcClassicLink",
    "ec2:DescribeVpcClassicLinkDnsSupport",
    "ec2:DescribeVpcEndpointConnectionNotifications",
    "ec2:DescribeVpcEndpointConnections",
    "ec2:DescribeVpcEndpointServiceConfigurations",
    "ec2:DescribeVpcEndpointServicePermissions",
    "ec2:DescribeVpcEndpointServices",
    "ec2:DescribeVpcEndpoints",
    "ec2:DescribeVpcPeeringConnections",
    "ec2:DescribeVpcs",
    "ec2:DescribeVpnGateways",
    "ec2:ExportClientVpnClientCertificateRevocationList",
    "ec2:ExportClientVpnClientConfiguration",
    "ec2:GetGroupsForCapacityReservation",
    "ec2:GetTransitGatewayAttachmentPropagations",
    "ec2:GetTransitGatewayMulticastDomainAssociations",
    "ec2:GetTransitGatewayPrefixListReferences",
    "ec2:GetTransitGatewayRouteTableAssociations",
    "ec2:GetTransitGatewayRouteTablePropagations",
    "ec2:SearchLocalGatewayRoutes",
    "ec2:SearchTransitGatewayMulticastGroups",
    "ec2:SearchTransitGatewayRoutes"
  ],
  "Resource": "*",
  "Effect": "Allow"
}

```

allReadActions

Adds all actions with access level **read** to the statement.

JavaScript

Python

Result

```
new statement Ec2() //
  allow()
  allReadActions()
```

```
statement.Ec2() \
  .allow() \
  .all_read_actions()
```

```
{
  "Action": [
    "ec2:DescribeElasticGpus",
    "ec2:DescribeFastSnapshotRestores",
    "ec2:DescribeScheduledInstanceAvailability",
    "ec2:DescribeScheduledInstances",
    "ec2:DescribeTags",
    "ec2:DescribeVolumesModifications",
    "ec2:DescribeVpnConnections",
    "ec2:GetAssociatedEnclaveCertificateIamRoles",
    "ec2:GetAssociatedIpv6PoolCidrs",
    "ec2:GetCapacityReservationUsage",
    "ec2:GetCoipPoolUsage",
    "ec2:GetConsoleOutput",
    "ec2:GetConsoleScreenshot",
    "ec2:GetDefaultCreditSpecification",
    "ec2:GetEbsDefaultKmsKeyId",
    "ec2:GetEbsEncryptionByDefault",
    "ec2:GetHostReservationPurchasePreview",
    "ec2:GetLaunchTemplateData",
    "ec2:GetManagedPrefixListAssociations",
    "ec2:GetManagedPrefixListEntries",
    "ec2:GetPasswordData",
    "ec2:GetReservedInstancesExchangeQuote",
    "ec2:GetSerialConsoleAccessStatus"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

allWriteActions

Adds all actions with access level **write** to the statement.

JavaScript

Python

Result

```
new statement Ec2() //
  allow()
  allWriteActions()
```

```
statement.Ec2() \
  .allow() \
  .all_write_actions()
```

```
{
  "Action": [
    "ec2:AcceptReservedInstancesExchangeQuote",
    "ec2:AcceptTransitGatewayMulticastDomainAssociations",
    "ec2:AcceptTransitGatewayPeeringAttachment",
    "ec2:AcceptTransitGatewayVpcAttachment",
    "ec2:AcceptVpcEndpointConnections",
    "ec2:AcceptVpcPeeringConnection",
    "ec2:AdvertiseByoipCidr",
    "ec2:AllocateAddress",
    "ec2:AllocateHosts",
    "ec2:ApplySecurityGroupsToClientVpnTargetNetwork",
    "ec2:AssignIpv6Addresses",
    "ec2:AssignPrivateIpAddresses",
    "ec2:AssociateAddress",
    "ec2:AssociateClientVpnTargetNetwork",
    "ec2:AssociateDhcpOptions",
    "ec2:AssociateEnclaveCertificateIamRole",
    "ec2:AssociateIamInstanceProfile",
    "ec2:AssociateRouteTable",
    "ec2:AssociateSubnetCidrBlock",
    "ec2:AssociateTransitGatewayMulticastDomain",
    "ec2:AssociateTransitGatewayRouteTable",
    "ec2:AssociateVpcCidrBlock",
    "ec2:AttachClassicLinkVpc",
    "ec2:AttachInternetGateway",
    "ec2:AttachNetworkInterface",
    "ec2:AttachVolume",
    "ec2:AttachVpnGateway",
    "ec2:AuthorizeClientVpnIngress",
    "ec2:AuthorizeSecurityGroupEgress",
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:BundleInstance",
    "ec2:CancelBundleTask",
    "ec2:CancelCapacityReservation",
    "ec2:CancelConversionTask",
    "ec2:CancelExportTask",
    "ec2:CancelImportTask",
    "ec2:CancelReservedInstancesListing",
    "ec2:CancelSpotFleetRequests",
```

(continues on next page)

(continued from previous page)

```

"ec2:CancelSpotInstanceRequests",
"ec2:ConfirmProductInstance",
"ec2:CopyFpgaImage",
"ec2:CopyImage",
"ec2:CopySnapshot",
"ec2:CreateCapacityReservation",
"ec2:CreateCarrierGateway",
"ec2:CreateClientVpnEndpoint",
"ec2:CreateClientVpnRoute",
"ec2:CreateCustomerGateway",
"ec2:CreateDefaultSubnet",
"ec2:CreateDefaultVpc",
"ec2:CreateDhcpOptions",
"ec2:CreateEgressOnlyInternetGateway",
"ec2:CreateFleet",
"ec2:CreateFlowLogs",
"ec2:CreateFpgaImage",
"ec2:CreateImage",
"ec2:CreateInstanceExportTask",
"ec2:CreateInternetGateway",
"ec2:CreateKeyPair",
"ec2:CreateLaunchTemplate",
"ec2:CreateLaunchTemplateVersion",
"ec2:CreateLocalGatewayRoute",
"ec2:CreateLocalGatewayRouteTableVpcAssociation",
"ec2:CreateManagedPrefixList",
"ec2:CreateNatGateway",
"ec2:CreateNetworkAcl",
"ec2:CreateNetworkAclEntry",
"ec2:CreateNetworkInsightsPath",
"ec2:CreateNetworkInterface",
"ec2:CreatePlacementGroup",
"ec2:CreateReservedInstancesListing",
"ec2:CreateRoute",
"ec2:CreateRouteTable",
"ec2:CreateSecurityGroup",
"ec2:CreateSnapshot",
"ec2:CreateSnapshots",
"ec2:CreateSpotDatafeedSubscription",
"ec2:CreateSubnet",
"ec2:CreateTrafficMirrorFilter",
"ec2:CreateTrafficMirrorFilterRule",
"ec2:CreateTrafficMirrorSession",
"ec2:CreateTrafficMirrorTarget",
"ec2:CreateTransitGateway",
"ec2:CreateTransitGatewayConnect",
"ec2:CreateTransitGatewayConnectPeer",
"ec2:CreateTransitGatewayMulticastDomain",
"ec2:CreateTransitGatewayPeeringAttachment",
"ec2:CreateTransitGatewayPrefixListReference",
"ec2:CreateTransitGatewayRoute",
"ec2:CreateTransitGatewayRouteTable",
"ec2:CreateTransitGatewayVpcAttachment",
"ec2:CreateVolume",
"ec2:CreateVpc",
"ec2:CreateVpcEndpoint",
"ec2:CreateVpcEndpointConnectionNotification"

```

(continues on next page)

(continued from previous page)

```

"ec2:CreateVpcEndpointServiceConfiguration",
"ec2:CreateVpcPeeringConnection",
"ec2:CreateVpnConnection",
"ec2:CreateVpnConnectionRoute",
"ec2:CreateVpnGateway",
"ec2:DeleteCarrierGateway",
"ec2:DeleteClientVpnEndpoint",
"ec2:DeleteClientVpnRoute",
"ec2:DeleteCustomerGateway",
"ec2:DeleteDhcpOptions",
"ec2:DeleteEgressOnlyInternetGateway",
"ec2:DeleteFleets",
"ec2:DeleteFlowLogs",
"ec2:DeleteFpgaImage",
"ec2:DeleteInternetGateway",
"ec2:DeleteKeyPair",
"ec2:DeleteLaunchTemplate",
"ec2:DeleteLaunchTemplateVersions",
"ec2:DeleteLocalGatewayRoute",
"ec2:DeleteLocalGatewayRouteTableVpcAssociation",
"ec2:DeleteManagedPrefixList",
"ec2:DeleteNatGateway",
"ec2:DeleteNetworkAcl",
"ec2:DeleteNetworkAclEntry",
"ec2:DeleteNetworkInsightsAnalysis",
"ec2:DeleteNetworkInsightsPath",
"ec2:DeleteNetworkInterface",
"ec2:DeletePlacementGroup",
"ec2:DeleteQueuedReservedInstances",
"ec2:DeleteRoute",
"ec2:DeleteRouteTable",
"ec2:DeleteSecurityGroup",
"ec2:DeleteSnapshot",
"ec2:DeleteSpotDatafeedSubscription",
"ec2:DeleteSubnet",
"ec2:DeleteTrafficMirrorFilter",
"ec2:DeleteTrafficMirrorFilterRule",
"ec2:DeleteTrafficMirrorSession",
"ec2:DeleteTrafficMirrorTarget",
"ec2:DeleteTransitGateway",
"ec2:DeleteTransitGatewayConnect",
"ec2:DeleteTransitGatewayConnectPeer",
"ec2:DeleteTransitGatewayMulticastDomain",
"ec2:DeleteTransitGatewayPeeringAttachment",
"ec2:DeleteTransitGatewayPrefixListReference",
"ec2:DeleteTransitGatewayRoute",
"ec2:DeleteTransitGatewayRouteTable",
"ec2:DeleteTransitGatewayVpcAttachment",
"ec2:DeleteVolume",
"ec2:DeleteVpc",
"ec2:DeleteVpcEndpointConnectionNotifications",
"ec2:DeleteVpcEndpointServiceConfigurations",
"ec2:DeleteVpcEndpoints",
"ec2:DeleteVpcPeeringConnection",
"ec2:DeleteVpnConnection",
"ec2:DeleteVpnConnectionRoute",
"ec2:DeleteVpnGateway",

```

(continues on next page)

(continued from previous page)

```

"ec2:DeprovisionByoipCidr",
"ec2:DeregisterImage",
"ec2:DeregisterInstanceEventNotificationAttributes",
"ec2:DeregisterTransitGatewayMulticastGroupMembers",
"ec2:DeregisterTransitGatewayMulticastGroupSources",
"ec2:DetachClassicLinkVpc",
"ec2:DetachInternetGateway",
"ec2:DetachNetworkInterface",
"ec2:DetachVolume",
"ec2:DetachVpnGateway",
"ec2:DisableEbsEncryptionByDefault",
"ec2:DisableFastSnapshotRestores",
"ec2:DisableSerialConsoleAccess",
"ec2:DisableTransitGatewayRouteTablePropagation",
"ec2:DisableVgwRoutePropagation",
"ec2:DisableVpcClassicLink",
"ec2:DisableVpcClassicLinkDnsSupport",
"ec2:DisassociateAddress",
"ec2:DisassociateClientVpnTargetNetwork",
"ec2:DisassociateEnclaveCertificateIamRole",
"ec2:DisassociateIamInstanceProfile",
"ec2:DisassociateRouteTable",
"ec2:DisassociateSubnetCidrBlock",
"ec2:DisassociateTransitGatewayMulticastDomain",
"ec2:DisassociateTransitGatewayRouteTable",
"ec2:DisassociateVpcCidrBlock",
"ec2:EnableEbsEncryptionByDefault",
"ec2:EnableFastSnapshotRestores",
"ec2:EnableSerialConsoleAccess",
"ec2:EnableTransitGatewayRouteTablePropagation",
"ec2:EnableVgwRoutePropagation",
"ec2:EnableVolumeIO",
"ec2:EnableVpcClassicLink",
"ec2:EnableVpcClassicLinkDnsSupport",
"ec2:ExportImage",
"ec2:ExportTransitGatewayRoutes",
"ec2:ImportClientVpnClientCertificateRevocationList",
"ec2:ImportImage",
"ec2:ImportInstance",
"ec2:ImportKeyPair",
"ec2:ImportSnapshot",
"ec2:ImportVolume",
"ec2:ModifyAddressAttribute",
"ec2:ModifyAvailabilityZoneGroup",
"ec2:ModifyCapacityReservation",
"ec2:ModifyClientVpnEndpoint",
"ec2:ModifyDefaultCreditSpecification",
"ec2:ModifyEbsDefaultKmsKeyId",
"ec2:ModifyFleet",
"ec2:ModifyFpgaImageAttribute",
"ec2:ModifyHosts",
"ec2:ModifyIdFormat",
"ec2:ModifyIdentityIdFormat",
"ec2:ModifyImageAttribute",
"ec2:ModifyInstanceAttribute",
"ec2:ModifyInstanceCapacityReservationAttributes",
"ec2:ModifyInstanceCreditSpecification".

```

(continues on next page)

(continued from previous page)

```

"ec2:ModifyInstanceEventStartTime",
"ec2:ModifyInstanceMetadataOptions",
"ec2:ModifyInstancePlacement",
"ec2:ModifyLaunchTemplate",
"ec2:ModifyManagedPrefixList",
"ec2:ModifyNetworkInterfaceAttribute",
"ec2:ModifyReservedInstances",
"ec2:ModifySpotFleetRequest",
"ec2:ModifySubnetAttribute",
"ec2:ModifyTrafficMirrorFilterNetworkServices",
"ec2:ModifyTrafficMirrorFilterRule",
"ec2:ModifyTrafficMirrorSession",
"ec2:ModifyTransitGateway",
"ec2:ModifyTransitGatewayPrefixListReference",
"ec2:ModifyTransitGatewayVpcAttachment",
"ec2:ModifyVolume",
"ec2:ModifyVolumeAttribute",
"ec2:ModifyVpcAttribute",
"ec2:ModifyVpcEndpoint",
"ec2:ModifyVpcEndpointConnectionNotification",
"ec2:ModifyVpcEndpointServiceConfiguration",
"ec2:ModifyVpcPeeringConnectionOptions",
"ec2:ModifyVpcTenancy",
"ec2:ModifyVpnConnection",
"ec2:ModifyVpnConnectionOptions",
"ec2:ModifyVpnTunnelCertificate",
"ec2:ModifyVpnTunnelOptions",
"ec2:MonitorInstances",
"ec2:MoveAddressToVpc",
"ec2:ProvisionByoipCidr",
"ec2:PurchaseHostReservation",
"ec2:PurchaseReservedInstancesOffering",
"ec2:PurchaseScheduledInstances",
"ec2:RebootInstances",
"ec2:RegisterImage",
"ec2:RegisterInstanceEventNotificationAttributes",
"ec2:RegisterTransitGatewayMulticastGroupMembers",
"ec2:RegisterTransitGatewayMulticastGroupSources",
"ec2:RejectTransitGatewayMulticastDomainAssociations",
"ec2:RejectTransitGatewayPeeringAttachment",
"ec2:RejectTransitGatewayVpcAttachment",
"ec2:RejectVpcEndpointConnections",
"ec2:RejectVpcPeeringConnection",
"ec2:ReleaseAddress",
"ec2:ReleaseHosts",
"ec2:ReplaceIamInstanceProfileAssociation",
"ec2:ReplaceNetworkAclAssociation",
"ec2:ReplaceNetworkAclEntry",
"ec2:ReplaceRoute",
"ec2:ReplaceRouteTableAssociation",
"ec2:ReplaceTransitGatewayRoute",
"ec2:ReportInstanceStatus",
"ec2:RequestSpotFleet",
"ec2:RequestSpotInstances",
"ec2:ResetAddressAttribute",
"ec2:ResetEbsDefaultKmsKeyId",
"ec2:ResetFpgaImageAttribute",

```

(continues on next page)

(continued from previous page)

```

        "ec2:ResetImageAttribute",
        "ec2:ResetInstanceAttribute",
        "ec2:ResetNetworkInterfaceAttribute",
        "ec2:RestoreAddressToClassic",
        "ec2:RestoreManagedPrefixListVersion",
        "ec2:RevokeClientVpnIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:RunInstances",
        "ec2:RunScheduledInstances",
        "ec2:SendDiagnosticInterrupt",
        "ec2:StartInstances",
        "ec2:StartNetworkInsightsAnalysis",
        "ec2:StartVpcEndpointServicePrivateDnsVerification",
        "ec2:StopInstances",
        "ec2:TerminateClientVpnConnections",
        "ec2:TerminateInstances",
        "ec2:UnassignIpv6Addresses",
        "ec2:UnassignPrivateIpAddresses",
        "ec2:UnmonitorInstances",
        "ec2:UpdateSecurityGroupRuleDescriptionsEgress",
        "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
        "ec2:WithdrawByoipCidr"
    ],
    "Resource": "*",
    "Effect": "Allow"
}

```

allPermissionManagementActions

Adds all actions with [access level permission management](#) to the statement.

JavaScript

Python

Result

```

new statement Ec2() //
  allow()
  allPermissionManagementActions()

```

```

statement.Ec2() \
  .allow() \
  .all_permission_management_actions()

```

```

{
  "Action": [
    "ec2:CreateNetworkInterfacePermission",
    "ec2:DeleteNetworkInterfacePermission",
    "ec2:ModifySnapshotAttribute",
    "ec2:ModifyVpcEndpointServicePermissions",
    "ec2:ResetSnapshotAttribute"
  ],
  "Resource": "*"
}

```

(continues on next page)

(continued from previous page)

```
"Effect": "Allow"
}
```

allTaggingActions

Adds all actions with access level **tagging** to the statement.

JavaScript

Python

Result

```
new statement Ec2 () //
  allow ()
  allTaggingActions ()
```

```
statement.Ec2 () \
  .allow () \
  .all_tagging_actions ()
```

```
{
  "Action": [
    "ec2:CreateTags",
    "ec2:DeleteTags"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

2.4 if (Condition)

Every available IAM **condition** key is represented by a distinct method. These methods start with **if**. You allow/deny something **if** a condition is met.

Every statement provider (e.g. `Ec2`) brings its unique conditions. **Global condition context keys** start with `ifAws`.

Note: Multiple conditions on a statement all have to be true.

When you have multiple values on a single condition, one of them has to be true.

Other than that, IAM has no concept of OR. You need to define multiple statements for each OR branch.

JavaScript

Python

Result

```
new statement Ec2 ()
  allow ()
  toStartInstances ()
  ifEncrypted ()
```

(continues on next page)

(continued from previous page)

```

    ifInstanceType(['t3.micro', 't3.nano'])
    ifAssociatePublicIpAddress(false)
    ifAwsRequestTag('Owner', 'John')

```

```

statement.Ec2() \
    .allow() \
    .to_start_instances() \
    .if_encrypted() \
    .if_instance_type(['t3.micro', 't3.nano']) \
    .if_associate_public_ip_address(False) \
    .if_aws_request_tag('Owner', 'John')

```

```

{
  "Condition": {
    "Bool": {
      "ec2:Encrypted": "true",
      "ec2:AssociatePublicIpAddress": "false"
    },
    "StringLike": {
      "ec2:InstanceType": [
        "t3.micro",
        "t3.nano"
      ],
      "aws:RequestTag/Owner": "John"
    }
  },
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}

```

Every `if` method has a default operator. For instance, conditions which operate on strings usually have `StringLike` as default. Most methods allow you to pass an operator as last argument.

JavaScript

Python

Result

```

new statement.Ec2()
    allow()
    toStartInstances()
    ifAwsRequestTag('TagWithSpecialChars', '*John*', 'StringEquals')

```

```

statement.Ec2() \
    .allow() \
    .to_start_instances() \
    .if_aws_request_tag('TagWithSpecialChars', '*John*', 'StringEquals')

```

```

{
  "Condition": {
    "StringEquals": {
      "aws:RequestTag/TagWithSpecialChars": "*John*"
    }
  },
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}

```

(continues on next page)

(continued from previous page)

```

    "Action": "ec2:StartInstances",
    "Resource": "*",
    "Effect": "Allow"
  }

```

In case of **missing conditions**, you can define just any condition yourself via `if()`:

JavaScript

Python

Result

```

new statement.Ec2()
  allow()
  toStartInstances()
  if('ec2:missingCondition', 'some-value')

```

```

statement.Ec2() \
  .allow() \
  .to_start_instances() \
  .if_('ec2:missingCondition', 'some-value')

```

```

{
  "Condition": {
    "StringLike": {
      "ec2:missingCondition": "some-value"
    }
  },
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}

```

2.4.1 Operators

Condition operators can just be passed as strings. Or you can use the class `statement.Operator()`:

JavaScript

Python

Result

```

new statement.Dynamodb()
  allow()
  toGetItem()
  onTable('Thread')
  ifAttributes(
    ['ID', 'Message', 'Tags'],
    new statement.Operator().stringEquals().forAllValues()
  )

```

```

statement.Dynamodb() \
  .allow() \
  .to_get_item() \

```

(continues on next page)

(continued from previous page)

```
.on_table('Thread') \
.if_attributes(['ID', 'Message', 'Tags'],
               statement.Operator().string_equals().for_all_values())
```

```
{
  "Condition": {
    "ForAllValues:StringEquals": {
      "dynamodb:Attributes": {
        "ID",
        "Message",
        "Tags"
      }
    }
  },
  "Action": "dynamodb:GetItem",
  "Resource": "arn:aws:dynamodb:*:*:table/Thread",
  "Effect": "Allow"
}
```

JavaScript

Python

Result

```
new statement.Dynamodb()
  .deny()
  .toPutItem()
  .onTable('Thread')
  .ifAttributes(
    ['ID', 'PostDateTime'],
    new statement.Operator().stringEquals().forAnyValue()
  )
```

```
statement.Dynamodb() \
  .deny() \
  .to_put_item() \
  .on_table('Thread') \
  .if_attributes(['ID', 'PostDateTime'],
                 statement.Operator().string_equals().for_any_value())
```

```
{
  "Condition": {
    "ForAnyValue:StringEquals": {
      "dynamodb:Attributes": {
        "ID",
        "PostDateTime"
      }
    }
  },
  "Action": "dynamodb:PutItem",
  "Resource": "arn:aws:dynamodb:*:*:table/Thread",
  "Effect": "Deny"
}
```

JavaScript

Python

Result

```
new statement Ec2()
  allow()
  toStartInstances()
  ifAwsRequestTag(
    'Environment',
    ['Production', 'Staging', 'Dev'],
    new statement.Operator().stringEquals().ifExists()
  )
```

```
statement.Ec2() \
  .allow() \
  .to_start_instances() \
  .if_aws_request_tag('Environment',
    ['Production', 'Staging', 'Dev'],
    statement.Operator().string_equals().if_exists())
```

```
{
  "Condition": {
    "StringEqualsIfExists": {
      "aws:RequestTag/Environment": [
        "Production",
        "Staging",
        "Dev"
      ]
    }
  },
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
}
```

2.5 on (Resource)

Every available IAM [resources](#) key is represented by a distinct method. These methods start with `on`. You allow/deny something **on** a specific resource (or pattern).

JavaScript

Python

Result

```
new statement S3()
  allow()
  allActions()
  onBucket('example-bucket')
  onObject('example-bucket', 'some/path/*')
```

```
statement.S3() \
  .allow() \
  .all_actions() \
```

(continues on next page)

(continued from previous page)

```
.on_bucket('example-bucket') \
.on_object('example-bucket', 'some/path/*')
```

```
{
  "Action": "s3:*",
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/some/path/*"
  ],
  "Effect": "Allow"
}
```

In case of [missing resources](#) or if you already have an ARN ready, use the `on()` method:

JavaScript

Python

Result

```
new statement.S3() //
  allow()
  allActions()
  on(
    'arn:aws:s3:::example-bucket', //
    'arn:aws:s3:::another-bucket'
  )
```

```
statement.S3() \
  .allow() \
  .all_actions() \
  .on('arn:aws:s3:::example-bucket',
      'arn:aws:s3:::another-bucket')
```

```
{
  "Action": "s3:*",
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::another-bucket"
  ],
  "Effect": "Allow"
}
```

If no resources are applied to the statement without principals, it defaults to all resources (*).

2.6 for (Principal)

Note: If you use the CDK variant of the package, don't attempt to create an assume policy with this package. Assume policies have to be of type `IPrincipal` and can easily be created with the [iam](#) package.

Every possible [principal](#) is represented by a distinct method. These methods start with `for`. You allow/deny something **for** a specific principal.

JavaScript

Python

Result

```
const s1 = new statement.Sts() //
    allow()
    toAssumeRole()
    forAccount('1234567890');

const s2 = new statement.Sts()
    allow()
    toAssumeRoleWithSAML()
    forService('lambda.amazonaws.com');

const s3 = new statement.Sts() //
    allow()
    toAssumeRole()
    forUser('1234567890', 'Bob');

const s4 = new statement.Sts() //
    allow()
    toAssumeRole()
    forRole('1234567890', 'role-name');

const s5 = new statement.Sts() //
    allow()
    toAssumeRoleWithSAML()
    forFederatedCognito();

const s6 = new statement.Sts() //
    allow()
    toAssumeRoleWithSAML()
    forFederatedAmazon();

const s7 = new statement.Sts() //
    allow()
    toAssumeRoleWithSAML()
    forFederatedGoogle();

const s8 = new statement.Sts() //
    allow()
    toAssumeRoleWithSAML()
    forFederatedFacebook();

const s9 = new statement.Sts()
    allow()
    toAssumeRoleWithSAML()
    forSaml('1234567890', 'saml-provider');

const s10 = new statement.Sts() //
    allow()
    toAssumeRole()
    forPublic();

const s11 = new statement.Sts()
    allow()
    toAssumeRole()
    forAssumedRoleSession('123456789', 'role-name', 'session-name');
```

(continues on next page)

(continued from previous page)

```
const s12 = new statement.Sts() //
    allow()
    toAssumeRole()
    forCanonicalUser('userID');

const s13 = new statement.Sts() //
    allow()
    toAssumeRole()
    for('arn:foo:bar');
```

```
s1 = statement.Sts() \
    .allow() \
    .to_assume_role() \
    .for_account('1234567890')

s2 = statement.Sts() \
    .allow() \
    .to_assume_role_with_saml() \
    .for_service('lambda.amazonaws.com')

s3 = statement.Sts() \
    .allow() \
    .to_assume_role() \
    .for_user('1234567890', 'Bob')

s4 = statement.Sts() \
    .allow() \
    .to_assume_role() \
    .for_role('1234567890', 'role-name')

s5 = statement.Sts() \
    .allow() \
    .to_assume_role_with_saml() \
    .for_federated_cognito()

s6 = statement.Sts() \
    .allow() \
    .to_assume_role_with_saml() \
    .for_federated_amazon()

s7 = statement.Sts() \
    .allow() \
    .to_assume_role_with_saml() \
    .for_federated_google()

s8 = statement.Sts() \
    .allow() \
    .to_assume_role_with_saml() \
    .for_federated_facebook()

s9 = statement.Sts() \
    .allow() \
    .to_assume_role_with_saml() \
    .for_saml('1234567890', 'saml-provider')
```

(continues on next page)

(continued from previous page)

```

s10 = statement.Sts() \
    .allow() \
    .to_assume_role() \
    .for_public()

s11 = statement.Sts() \
    .allow() \
    .to_assume_role() \
    .for_assumed_role_session('123456789', 'role-name', 'session-name')

s12 = statement.Sts() \
    .allow() \
    .to_assume_role() \
    .for_canonical_user('userID')

s13 = statement.Sts() \
    .allow() \
    .to_assume_role() \
    .for_('arn:foo:bar')

```

```

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::1234567890:root"
    ]
  }
}

{
  "Action": "sts:AssumeRoleWithSAML",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "lambda.amazonaws.com"
    ]
  }
}

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::1234567890:user/Bob"
    ]
  }
}

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::1234567890:role/role-name"
    ]
  }
}

```

(continues on next page)

(continued from previous page)

```

{
  "Action": "sts:AssumeRoleWithSAML",
  "Effect": "Allow",
  "Principal": {
    "Federated": [
      "cognito-identity.amazonaws.com"
    ]
  }
}

{
  "Action": "sts:AssumeRoleWithSAML",
  "Effect": "Allow",
  "Principal": {
    "Federated": [
      "www.amazon.com"
    ]
  }
}

{
  "Action": "sts:AssumeRoleWithSAML",
  "Effect": "Allow",
  "Principal": {
    "Federated": [
      "accounts.google.com"
    ]
  }
}

{
  "Action": "sts:AssumeRoleWithSAML",
  "Effect": "Allow",
  "Principal": {
    "Federated": [
      "graph.facebook.com"
    ]
  }
}

{
  "Action": "sts:AssumeRoleWithSAML",
  "Effect": "Allow",
  "Principal": {
    "Federated": [
      "arn:aws:iam::1234567890:saml-provider/saml-provider"
    ]
  }
}

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "*"
    ]
  }
}

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",

```

(continues on next page)

(continued from previous page)

```

    "Principal": {
      "AWS": [
        "arn:aws:sts::123456789:assumed-role/role-name/session-name"
      ]
    }
  }
}

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "CanonicalUser": [
      "userID"
    ]
  }
}

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:foo:bar"
    ]
  }
}

```

The CDK variant of the package has an additional method `forCdkPrincipal`, which takes any number of `iam.IPrincipal` objects:

JavaScript

Python

Result

```

new statement.Sts()
  allow()
  toAssumeRole()
  forCdkPrincipal(
    new iam.ServicePrincipal('sns.amazonaws.com'),
    new iam.ServicePrincipal('lambda.amazonaws.com')
  )

```

```

statement.Sts() \
  .allow() \
  .to_assume_role() \
  .for_cdk_principal(iam.ServicePrincipal('sns.amazonaws.com'),
    iam.ServicePrincipal('lambda.amazonaws.com'))

```

```

{
  "Action": "sts:AssumeRole",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "${Token[sns.amazonaws.com.9]}",
      "${Token[lambda.amazonaws.com.10]}"
    ]
  }
}

```

(continues on next page)

(continued from previous page)

```
}  
}
```

2.7 not (notAction, notResource and notPrincipal)

Warning: Make sure, you well understand the concepts of `notAction`, `notResource` and `notPrincipal`. This is where things quickly go wrong, especially when used in combination.

2.7.1 notActions

Switches the policy provider to use `NotAction`.

JavaScript

Python

Result

```
new statement.S3()  
  allow()  
  notActions()  
  toDeleteBucket()  
  onBucket('example-bucket')
```

```
statement.S3() \  
  .allow() \  
  .not_actions() \  
  .to_delete_bucket() \  
  .on_bucket('example-bucket')
```

```
{  
  "NotAction": "s3:DeleteBucket",  
  "Resource": "arn:aws:s3:::example-bucket",  
  "Effect": "Allow"  
}
```

2.7.2 notResources

Switches the policy provider to use `NotResource`.

JavaScript

Python

Result

```
new statement.S3()  
  allow()  
  notResources()  
  toDeleteBucket()  
  onBucket('example-bucket')
```

```
statement.S3() \
    .allow() \
    .not_resources() \
    .to_delete_bucket() \
    .on_bucket('example-bucket')
```

```
{
  "Action": "s3:DeleteBucket",
  "NotResource": "arn:aws:s3:::example-bucket",
  "Effect": "Allow"
}
```

2.7.3 notPrincipals

Switches the policy provider to use `NotPrincipal`.

JavaScript

Python

Result

```
new statement S3()
  allow()
  allActions()
  notPrincipals()
  forUser('1234567890', 'Bob')
  onObject('example-bucket', '*')
```

```
statement.S3() \
    .allow() \
    .all_actions() \
    .not_principals() \
    .for_user('1234567890', 'Bob')
    .on_object('example-bucket', '*')
```

```
{
  "Action": "s3:*",
  "Resource": "arn:aws:s3:::example-bucket/*",
  "Effect": "Allow",
  "NotPrincipal": {
    "AWS": {
      "arn:aws:iam::1234567890:user/Bob"
    }
  }
}
```

2.8 compact

This method can be used to convert a list of actions down to a list of wildcard patterns. This can be handy to reduce the policy size, especially when you work with *Access levels*.

Attention: When AWS later adds new actions, the patterns might match additional actions.

JavaScript

Python

Result

```
new statement Ec2() //
  allow()
  allReadActions()
  allListActions()
  compact()
```

```
statement.Ec2() \
  .allow() \
  .all_read_actions()
  .all_list_actions()
  .compact()
```

```
{
  "Action": [
    "ec2:Describe*",
    "ec2:ExportClientVpnClientC*",
    "ec2:Get*",
    "ec2:SearchLocalGatewayRoutes",
    "ec2:SearchTransitGateway*"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

EXAMPLES

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

JavaScript

Python

Result

```
const policy = {
  Version: '2012-10-17',
  Statement: [
    new statement.Ec2()
      .allow()
      .toStartInstances()
      .ifAwsRequestTag('Owner', '${aws:username}'),
    new statement.Ec2()
      .allow()
      .toStopInstances()
      .ifResourceTag('Owner', '${aws:username}'),
    new statement.Ec2() //
      .allow()
      .allListActions()
      .allReadActions(),
  ],
};
```

```
policy = {
  'Version': '2012-10-17',
  'Statement': [
    statement.Ec2()
      .allow()
      .to_start_instances()
      .if_aws_request_tag('Owner', '${aws:username}')
      .to_json(),
    statement.Ec2()
      .allow()
      .to_stop_instances()
      .if_resource_tag('Owner', '${aws:username}')
      .to_json(),
    statement.Ec2()
```

(continues on next page)

(continued from previous page)

```

        .allow()
        .all_list_actions()
        .all_read_actions()
        .to_json()
    }
}

```

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Owner": "${aws:username}"
        }
      },
      "Action": "ec2:StartInstances",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Condition": {
        "StringLike": {
          "ec2:ResourceTag/Owner": "${aws:username}"
        }
      },
      "Action": "ec2:StopInstances",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAddressesAttribute",
        "ec2:DescribeAggregateIdFormat",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeBundleTasks",
        "ec2:DescribeByoipCidrs",
        "ec2:DescribeCapacityReservations",
        "ec2:DescribeCarrierGateways",
        "ec2:DescribeClassicLinkInstances",
        "ec2:DescribeClientVpnAuthorizationRules",
        "ec2:DescribeClientVpnConnections",
        "ec2:DescribeClientVpnEndpoints",
        "ec2:DescribeClientVpnRoutes",
        "ec2:DescribeClientVpnTargetNetworks",
        "ec2:DescribeCoipPools",
        "ec2:DescribeConversionTasks",
        "ec2:DescribeCustomerGateways",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeEgressOnlyInternetGateways",
        "ec2:DescribeElasticGpus",
        "ec2:DescribeExportImageTasks",
        "ec2:DescribeExportTasks",
        "ec2:DescribeFastSnapshotRestores",

```

(continues on next page)

(continued from previous page)

```
"ec2:DescribeFleetHistory",
"ec2:DescribeFleetInstances",
"ec2:DescribeFleets",
"ec2:DescribeFlowLogs",
"ec2:DescribeFpgaImageAttribute",
"ec2:DescribeFpgaImages",
"ec2:DescribeHostReservationOfferings",
"ec2:DescribeHostReservations",
"ec2:DescribeHosts",
"ec2:DescribeIamInstanceProfileAssociations",
"ec2:DescribeIdFormat",
"ec2:DescribeIdentityIdFormat",
"ec2:DescribeImageAttribute",
"ec2:DescribeImages",
"ec2:DescribeImportImageTasks",
"ec2:DescribeImportSnapshotTasks",
"ec2:DescribeInstanceAttribute",
"ec2:DescribeInstanceCreditSpecifications",
"ec2:DescribeInstanceEventNotificationAttributes",
"ec2:DescribeInstanceStatus",
"ec2:DescribeInstanceTypeOfferings",
"ec2:DescribeInstanceTypes",
"ec2:DescribeInstances",
"ec2:DescribeInternetGateways",
"ec2:DescribeIpv6Pools",
"ec2:DescribeKeyPairs",
"ec2:DescribeLaunchTemplateVersions",
"ec2:DescribeLaunchTemplates",
"ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations",
"ec2:DescribeLocalGatewayRouteTableVpcAssociations",
"ec2:DescribeLocalGatewayRouteTables",
"ec2:DescribeLocalGatewayVirtualInterfaceGroups",
"ec2:DescribeLocalGatewayVirtualInterfaces",
"ec2:DescribeLocalGateways",
"ec2:DescribeManagedPrefixLists",
"ec2:DescribeMovingAddresses",
"ec2:DescribeNatGateways",
"ec2:DescribeNetworkAcls",
"ec2:DescribeNetworkInsightsAnalyses",
"ec2:DescribeNetworkInsightsPaths",
"ec2:DescribeNetworkInterfaceAttribute",
"ec2:DescribeNetworkInterfacePermissions",
"ec2:DescribeNetworkInterfaces",
"ec2:DescribePlacementGroups",
"ec2:DescribePrefixLists",
"ec2:DescribePrincipalIdFormat",
"ec2:DescribePublicIpv4Pools",
"ec2:DescribeRegions",
"ec2:DescribeReservedInstances",
"ec2:DescribeReservedInstancesListings",
"ec2:DescribeReservedInstancesModifications",
"ec2:DescribeReservedInstancesOfferings",
"ec2:DescribeRouteTables",
"ec2:DescribeScheduledInstanceAvailability",
"ec2:DescribeScheduledInstances",
"ec2:DescribeSecurityGroupReferences",
"ec2:DescribeSecurityGroups"
```

(continues on next page)

(continued from previous page)

```

"ec2:DescribeSnapshotAttribute",
"ec2:DescribeSnapshots",
"ec2:DescribeSpotDatafeedSubscription",
"ec2:DescribeSpotFleetInstances",
"ec2:DescribeSpotFleetRequestHistory",
"ec2:DescribeSpotFleetRequests",
"ec2:DescribeSpotInstanceRequests",
"ec2:DescribeSpotPriceHistory",
"ec2:DescribeStaleSecurityGroups",
"ec2:DescribeSubnets",
"ec2:DescribeTags",
"ec2:DescribeTrafficMirrorFilters",
"ec2:DescribeTrafficMirrorSessions",
"ec2:DescribeTrafficMirrorTargets",
"ec2:DescribeTransitGatewayAttachments",
"ec2:DescribeTransitGatewayConnectPeers",
"ec2:DescribeTransitGatewayConnects",
"ec2:DescribeTransitGatewayMulticastDomains",
"ec2:DescribeTransitGatewayPeeringAttachments",
"ec2:DescribeTransitGatewayRouteTables",
"ec2:DescribeTransitGatewayVpcAttachments",
"ec2:DescribeTransitGateways",
"ec2:DescribeVolumeAttribute",
"ec2:DescribeVolumeStatus",
"ec2:DescribeVolumes",
"ec2:DescribeVolumesModifications",
"ec2:DescribeVpcAttribute",
"ec2:DescribeVpcClassicLink",
"ec2:DescribeVpcClassicLinkDnsSupport",
"ec2:DescribeVpcEndpointConnectionNotifications",
"ec2:DescribeVpcEndpointConnections",
"ec2:DescribeVpcEndpointServiceConfigurations",
"ec2:DescribeVpcEndpointServicePermissions",
"ec2:DescribeVpcEndpointServices",
"ec2:DescribeVpcEndpoints",
"ec2:DescribeVpcPeeringConnections",
"ec2:DescribeVpcs",
"ec2:DescribeVpnConnections",
"ec2:DescribeVpnGateways",
"ec2:ExportClientVpnClientCertificateRevocationList",
"ec2:ExportClientVpnClientConfiguration",
"ec2:GetAssociatedEnclaveCertificateIamRoles",
"ec2:GetAssociatedIpv6PoolCidrs",
"ec2:GetCapacityReservationUsage",
"ec2:GetCoipPoolUsage",
"ec2:GetConsoleOutput",
"ec2:GetConsoleScreenshot",
"ec2:GetDefaultCreditSpecification",
"ec2:GetEbsDefaultKmsKeyId",
"ec2:GetEbsEncryptionByDefault",
"ec2:GetGroupsForCapacityReservation",
"ec2:GetHostReservationPurchasePreview",
"ec2:GetLaunchTemplateData",
"ec2:GetManagedPrefixListAssociations",
"ec2:GetManagedPrefixListEntries",
"ec2:GetPasswordData",
"ec2:GetReservedInstancesExchangeQuote",

```

(continues on next page)

(continued from previous page)

```

        "ec2:GetSerialConsoleAccessStatus",
        "ec2:GetTransitGatewayAttachmentPropagations",
        "ec2:GetTransitGatewayMulticastDomainAssociations",
        "ec2:GetTransitGatewayPrefixListReferences",
        "ec2:GetTransitGatewayRouteTableAssociations",
        "ec2:GetTransitGatewayRouteTablePropagations",
        "ec2:SearchLocalGatewayRoutes",
        "ec2:SearchTransitGatewayMulticastGroups",
        "ec2:SearchTransitGatewayRoutes"
    ],
    "Resource": "*",
    "Effect": "Allow"
}
)

```

JavaScript

Python

Result

```

const policy = {
  Version: '2012-10-17',
  Statement: [
    new statement.Cloudformation() // allow all CFN actions
      .allow()
      .allActions(),
    new statement.All() // allow absolutely everything that is triggered via CFN
      .allow()
      .allActions()
      .ifAwsCalledVia('cloudformation.amazonaws.com'),
    new statement.S3() // allow access to the CDK staging bucket
      .allow()
      .allActions()
      .on('arn:aws:s3:::cdktoolkit-stagingbucket-*'),
    new statement.Account() // even when triggered via CFN, do not allow
↳modifications of the account
      .deny()
      .allPermissionManagementActions()
      .allWriteActions(),
    new statement.Organizations() // even when triggered via CFN, do not allow
↳modifications of the organization
      .deny()
      .allPermissionManagementActions()
      .allWriteActions(),
  ],
};

```

```

policy = {
  'Version': '2012-10-17',
  'Statement': [
    # allow all CFN actions
    statement.Cloudformation() \
      .allow() \
      .all_actions() \
      .to_json(),
  ],
}

```

(continues on next page)

(continued from previous page)

```

# allow access to the CDK staging bucket
statement.All() \
    .allow() \
    .all_actions() \
    .if_aws_called_via('cloudformation.amazonaws.com') \
    .to_json(),
# allow access to the CDK staging bucket
statement.S3() \
    .allow() \
    .all_actions() \
    .on('arn:aws:s3:::cdktoolkit-stagingbucket-*') \
    .to_json(),
# even when triggered via CFN, do not allow modifications of the
# account
statement.Account() \
    .deny() \
    .all_permission_management_actions() \
    .all_write_actions() \
    .to_json(),
# even when triggered via CFN, do not allow modifications of the
# organization
statement.Organizations() \
    .deny() \
    .all_permission_management_actions() \
    .all_write_actions() \
    .to_json()
}
)

```

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "cloudformation:*",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "cloudformation.amazonaws.com"
        }
      },
      "Action": "*",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::cdktoolkit-stagingbucket-*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "account:DisableRegion",

```

(continues on next page)

(continued from previous page)

```
        "account:EnableRegion"
    },
    "Resource": "*",
    "Effect": "Deny"
},
{
    "Action": [
        "organizations:AcceptHandshake",
        "organizations:AttachPolicy",
        "organizations:CancelHandshake",
        "organizations:CreateAccount",
        "organizations:CreateGovCloudAccount",
        "organizations:CreateOrganization",
        "organizations:CreateOrganizationalUnit",
        "organizations:CreatePolicy",
        "organizations:DeclineHandshake",
        "organizations>DeleteOrganization",
        "organizations>DeleteOrganizationalUnit",
        "organizations>DeletePolicy",
        "organizations:DeregisterDelegatedAdministrator",
        "organizations:DetachPolicy",
        "organizations:DisableAWSServiceAccess",
        "organizations:DisablePolicyType",
        "organizations:EnableAWSServiceAccess",
        "organizations:EnableAllFeatures",
        "organizations:EnablePolicyType",
        "organizations:InviteAccountToOrganization",
        "organizations:LeaveOrganization",
        "organizations:MoveAccount",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:RemoveAccountFromOrganization",
        "organizations:UpdateOrganizationalUnit",
        "organizations:UpdatePolicy"
    ],
    "Resource": "*",
    "Effect": "Deny"
}
]
```


COLLECTIONS

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

IAM Floyd provides commonly used statement collections. These can be called via:

JavaScript

Python

Result

```
new statement.Collection().allowEc2InstanceDeleteByOwner()
```

```
statements = statement.Collection().allow_ec2_instance_delete_by_owner()
```

```
{
  "Condition": {
    "StringLike": {
      "aws:RequestTag/Owner": "${aws:username}"
    }
  },
  "Action": "ec2:StartInstances",
  "Resource": "*",
  "Effect": "Allow"
},
{
  "Condition": {
    "StringLike": {
      "ec2:ResourceTag/Owner": "${aws:username}"
    }
  },
  "Action": "ec2:StopInstances",
  "Resource": "*",
  "Effect": "Allow"
}
```

Collections return a list of statements, which then can be used in a policy like this:

JavaScript

Python

Result

```
const policy = {
  Version: '2012-10-17',
  Statement: [
    ...new statement.Collection().allowEc2InstanceDeleteByOwner(),
  ],
};
```

```
statements = statement.Collection().allow_ec2_instance_delete_by_owner()
policy = {
  'Version': '2012-10-17',
  'Statement': list(map(lambda x: x.toJson(), statements)),
}
```

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "StringLike": {
          "aws:RequestTag/Owner": "${aws:username}"
        }
      },
      "Action": "ec2:StartInstances",
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Condition": {
        "StringLike": {
          "ec2:ResourceTag/Owner": "${aws:username}"
        }
      },
      "Action": "ec2:StopInstances",
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

4.1 Available collections

4.1.1 allowEc2InstanceDeleteByOwner

Allows stopping EC2 instance only for the user who started them.

FREQUENTLY ASKED QUESTIONS

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

5.1 Why should I use this package instead of writing policies by hand?

All actions, conditions and resource types of every service are explorable via code suggestion. The related documentation is available in the method description. In most cases you can avoid reading the documentation completely.

IntelliSense makes it super easy to find what you're looking for. But it also helps with discovering things you were not looking for! Users write more secure/restrictive policies because they can easily type `.if` and add conditions with a `<tab>` without looking up multiple documentation pages.

By calling methods of a class you protect yourself against typos. If your code doesn't compile/run because of a typo, you'll immediately notice. If instead you have a typo in your action list, IAM will silently accept your policy. You won't notice until you see a warning in the IAM console.

Allowing/Denying all actions based on `access level` is a feature AWS missed when designing IAM policies. With this package it is as easy as calling `.allWriteActions()`, `.allReadActions()` etc.

In IAM policies you can use wildcards to add actions to the statement. Wildcards often do not have enough power to define patterns and quickly include too many actions. This package enables you to select actions with regular expressions.

Limiting actions to specific resources via ARN is cumbersome. In this package, for every resource type there is a method, which not only helps with ARN creation - it also adds context to the code which helps to understand the meaning. The classical example here is to allow all actions on an S3 bucket and its containing objects:

```
{
  "Effect": "Allow",
  "Action": "s3:*",
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/some/path/*"
  ]
}
```

The first resource element is for the bucket itself. The second element is for the contained objects.

A beginner might make the mistake to think the first *or* the last entry is superfluous and remove it. This package has distinct methods to limit actions to a bucket and/or objects:

JavaScript

Python

Result

```
new statement.S3()
  .allow()
  .allActions()
  .onBucket('example-bucket')
  .onObject('example-bucket', 'some/path/*')
```

```
statement.S3() \
  .allow() \
  .all_actions() \
  .on_bucket('example-bucket') \
  .on_object('example-bucket', 'some/path/*')
```

```
{
  "Action": "s3:*",
  "Resource": [
    "arn:aws:s3:::example-bucket",
    "arn:aws:s3:::example-bucket/some/path/*"
  ],
  "Effect": "Allow"
}
```

And yes, it's shorter too.

5.2 Are all actions / conditions / resource types covered?

The code of IAM Floyd is generated from the [AWS Documentation](#). This means, **everything that was documented is covered**. Unfortunately not everything is documented. Users have repeatedly [reported](#) missing actions/conditions/resource types on the documentation repository.

If you believe something is missing, feel free to report it in the [IAM Floyd repository](#) or directly on the [AWS Documentation repository](#).

5.3 How often will there be updates to reflect IAM changes?

Once per day, at 2am UTC, the [AWS Documentation](#) is checked for updates. If anything changes, a new package will be released immediately.

5.4 Do you release new packages when a new CDK version is released?

No. I believe it's a myth and a user error if packages are incompatible with new releases of the CDK. `cdk-iam-floyd` is based on `cdk ^1.30.0` and so far I have not seen any issues.

5.5 Is the package following semantic versioning?

Mostly. For manual changes by developers this package follows [semver](#).

Automatic releases triggered by changes in the IAM documentation will always result in a minor update.

It has been observed that IAM actions have been [deleted](#) or [renamed](#). This case will not be reflected by a major update! If you had been using such a method your code will break. On the other hand, your code probably already is broken, since it creates a policy with invalid actions until you update to the latest release.

5.6 I don't like method chaining!

That's not a question. But yes, you can completely avoid method chaining:

JavaScript

Python

Result

```
const myStatement = new statement.Ec2();
myStatement.allow();
myStatement.toStartInstances();
myStatement.toStopInstances();
```

```
my_statement = statement.Ec2()
my_statement.allow()
my_statement.to_start_instances()
my_statement.to_stop_instances()
```

```
{
  "Action": [
    "ec2:StartInstances",
    "ec2:StopInstances"
  ],
  "Resource": "*",
  "Effect": "Allow"
}
```

5.7 Floyd?

George Floyd has been murdered by racist police officers on May 25, 2020.

This package is not named after him to just remind you of him and his death. I want this package to be of great help to you and I want you to use it on a daily base. Every time you use it, I want you to remember our society is ill and needs change. The riots will stop. The news will fade. The issue persists!

If this statement annoys you, this package is not for you.

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

The code contained in the `lib/generated` folder is generated from the [AWS documentation](#). The class- and function-names and their description therefore are property of AWS.

AWS and their services are trademarks, registered trademarks or trade dress of AWS in the U.S. and/or other countries.

This project is not affiliated, funded, or in any way associated with AWS.

6.1 License

IAM Floyd is licensed under [Apache License 2.0](#).

Dependencies might be released under different licenses. Especially the bundled packages `regex-parser` and `common-substrings` are released under the MIT License.

IAM FLOYD

Attention: This is an early version of the package. The API might change when new features are implemented. Therefore make sure you use an exact version in your `package.json/requirements.txt` before it reaches 1.0.0.

AWS IAM policy statement generator with fluent interface.

Support for:

- 265 Services
- 9849 Actions
- 1033 Resource Types
- 1043 Condition keys

7.1 Similar projects

- [cdk-iam-actions](#)
- [cdk-iam-generator](#)
- [iam-policy-generator](#)
- [policyuniverse](#)
- [policy_sentry](#)