IAM Floyd

Release 0.641.0

Daniel Schroeder

May 16, 2024

CONTENTS

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:

Starting with version 0.300.0, the packages are compatible with CDK v2. For CDK v1 you can use any version up to:

Find them all on libraries.io.

CHAPTER

ONE

CDK COMPATIBILITY MATRIX

CDK	cdk-iam-floyd
<= 1.151.0	<= 0.285.0
>= 1.152.0	0.286.0
>= 1.158.0	**No compatible version!**
>= 2.0.0	>= 0.300.0
>= 2.20.0	>= 0.351.0
>= 2.26.0	>= 0.377.0
2.29.x	**No compatible version!**
>= 2.30.0	>= 0.391.0

CHAPTER

TWO

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`:

```
# 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
```

TypeScript

JavaScript

```
// for use without AWS CDK use the iam-floyd package
import { Statement } from 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
import { Statement } from 'cdk-iam-floyd'
```

```
// for use without AWS CDK use the iam-floyd package
const { Statement } = require 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
const { Statement } = require 'cdk-iam-floyd'
```

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

Result

```
new Statement.Ec2().toStartInstances()
```

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

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

JavaScript

Result

```
new Statement.Ec2()
    .toStartInstances()
    .toStopInstances()
```

```
{
    "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

Result

```
new Statement.Ec2()
    .allow()
    .toStartInstances()
    .toStopInstances()
```

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

Or `deny()`:

JavaScript

Result

```
new Statement.Ec2()
    .deny()
    .toStartInstances()
    .toStopInstances()
```

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

To set an SID you can pass it as argument in the statement provider:

JavaScript

Result

```
new Statement.Ec2('MYSID', //
    .allow()
    .toStartInstances()
    .toStopInstances())
```

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

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

JavaScript

- `allListActions()`
- `allReadActions()`
- `allWriteActions()`
- `allPermissionManagementActions()`
- `allTaggingActions()`

JavaScript

Result

```
const s1 = new Statement.S3(//
    .deny()
    .allPermissionManagementActions())

const s2 = new Statement.S3(//
    .allow()
    .allListActions()
    .allReadActions())
```

```
{
    "Action": [
        "s3:BypassGovernanceRetention",
        "s3:DeleteAccessPointPolicy",
        "s3:DeleteAccessPointPolicyForObjectLambda",
        "s3:DeleteBucketPolicy",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:PutAccessPointPolicy",
        "s3:PutAccessPointPolicyForObjectLambda",
        "s3:PutAccessPointPublicAccessBlock",
        "s3:PutAccountPublicAccessBlock",
        "s3:PutBucketAcl",
        "s3:PutBucketPolicy",
        "s3:PutBucketPublicAccessBlock",
        "s3:PutMultiRegionAccessPointPolicy",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
    ],
    "Resource": "*",
    "Effect": "Deny"
},
{
    "Action": [
        "s3:DescribeJob",
        "s3:DescribeMultiRegionAccessPointOperation",
        "s3:GetAccelerateConfiguration",
        "s3:GetAccessGrant",
        "s3:GetAccessGrantsInstance",
        "s3:GetAccessGrantsInstanceForPrefix",
        "s3:GetAccessGrantsInstanceResourcePolicy",
        "s3:GetAccessGrantsLocation",
        "s3:GetAccessPoint",
        "s3:GetAccessPointConfigurationForObjectLambda",
        "s3:GetAccessPointForObjectLambda",
        "s3:GetAccessPointPolicy",
        "s3:GetAccessPointPolicyForObjectLambda",
        "s3:GetAccessPointPolicyStatus",
        "s3:GetAccessPointPolicyStatusForObjectLambda",
        "s3:GetAccountPublicAccessBlock",
        "s3:GetAnalyticsConfiguration",
        "s3:GetBucketAcl",
        "s3:GetBucketCORS",
        "s3:GetBucketLocation",
        "s3:GetBucketLogging",
        "s3:GetBucketNotification",
        "s3:GetBucketObjectLockConfiguration",
        "s3:GetBucketOwnershipControls",
        "s3:GetBucketPolicy",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPublicAccessBlock",
        "s3:GetBucketRequestPayment",
        "s3:GetBucketTagging",
        "s3:GetBucketVersioning",
```

(continues on next page)

(continued from previous page)

```
        "s3:GetBucketWebsite",
        "s3:GetDataAccess",
        "s3:GetEncryptionConfiguration",
        "s3:GetIntelligentTieringConfiguration",
        "s3:GetInventoryConfiguration",
        "s3:GetJobTagging",
        "s3:GetLifecycleConfiguration",
        "s3:GetMetricsConfiguration",
        "s3:GetMultiRegionAccessPoint",
        "s3:GetMultiRegionAccessPointPolicy",
        "s3:GetMultiRegionAccessPointPolicyStatus",
        "s3:GetMultiRegionAccessPointRoutes",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectAttributes",
        "s3:GetObjectLegalHold",
        "s3:GetObjectRetention",
        "s3:GetObjectTagging",
        "s3:GetObjectTorrent",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionAttributes",
        "s3:GetObjectVersionForReplication",
        "s3:GetObjectVersionTagging",
        "s3:GetObjectVersionTorrent",
        "s3:GetReplicationConfiguration",
        "s3:GetStorageLensConfiguration",
        "s3:GetStorageLensConfigurationTagging",
        "s3:GetStorageLensDashboard",
        "s3:GetStorageLensGroup",
        "s3:ListAccessGrants",
        "s3:ListAccessGrantsInstances",
        "s3:ListAccessGrantsLocations",
        "s3:ListAccessPoints",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
        "s3:ListMultipartUploadParts",
        "s3:ListStorageLensConfigurations",
        "s3:ListStorageLensGroups",
        "s3:ListTagsForResource"
    ],
    "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

Result

```
new Statement.Ec2()
  .deny()
  .allMatchingActions('/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:GetVpnConnectionDeviceSampleConfiguration",
        "ec2:GetVpnConnectionDeviceTypes",
        "ec2:GetVpnTunnelReplacementStatus",
        "ec2:ImportClientVpnClientCertificateRevocationList",
        "ec2:ModifyClientVpnEndpoint",
        "ec2:ModifyVpnConnection",
        "ec2:ModifyVpnConnectionOptions",
        "ec2:ModifyVpnTunnelCertificate",
        "ec2:ModifyVpnTunnelOptions",
        "ec2:ReplaceVpnTunnel",
        "ec2:RevokeClientVpnIngress",
        "ec2:TerminateClientVpnConnections"
    ],
    "Resource": "*"
```

(continues on next page)

(continued from previous page)

```
    "Effect": "Deny"
}
```

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

JavaScript

Result

```
new Statement.Ec2()
    .allow()
    .allActions()
```

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

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

JavaScript

Result

```
new Statement.Ec2()
    .allow()
    .toStartInstances()
    .ifEncrypted()
    .ifInstanceType('t3.micro', 't3.nano')
    .ifAssociatePublicIpAddress(false)
    .ifAwsRequestTag('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

Result

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

```
{
    "Condition": {
        "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

Result

```
new Statement.Ec2()
   .allow()
   .toStartInstances()
   .ifAwsRequestTag('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

Result

```
new Statement.S3()
   .allow()
   .allActions()
   .onBucket('example-bucket')
   .onObject('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

Result

```
new Statement.S3()
    .allow()
    .allActions()
    .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 `notAction()`, `notResource()` and `notPrincipal()`:

JavaScript

Result

```
new Statement.S3()
    .allow()
    .notAction()
    .toDeleteBucket()
    .onBucket('example-bucket')
```

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

JavaScript

Result

```
new Statement.S3()
   .allow()
   .notResource()
   .toDeleteBucket()
   .onBucket('example-bucket')
```

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

JavaScript

Result

```
new Statement.S3()
   .deny()
   .allActions()
   .notPrincipal()
   .forUser('1234567890', 'Bob')
   .onObject('example-bucket', '*')
```

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

CHAPTER

THREE

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.

3.1 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

Result

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

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

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

3.2 to (Action)

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

JavaScript

Result

```
new Statement.Ec2(//
   .allow()
   .toStartInstances()
   .toStopInstances())
```

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

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

JavaScript

Result

```
new Statement.Ec2(//
   .allow()
   .to('missingAction'))
```

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

3.3 all (Action)

While methods starting with `to` add a single action to a statement, methods starting with `all` add multiple actions.

3.3.1 allActions

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

JavaScript

Result

```
new Statement Ec2 //
   allow
   allActions
```

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

3.3.2 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

Result

```
new Statement Ec2 //
   deny
   allMatchingActions '/vpn/i'
```

```
{
   "Action"
      "ec2:ApplySecurityGroupsToClientVpnTargetNetwork"
      "ec2:AssociateClientVpnTargetNetwork"
      "ec2:AttachVpnGateway"
      "ec2:AuthorizeClientVpnIngress"
      "ec2:CreateClientVpnEndpoint"
      "ec2:CreateClientVpnRoute"
      "ec2:CreateVpnConnection"
      "ec2:CreateVpnConnectionRoute"
      "ec2:CreateVpnGateway"
```

(continues on next page)

(continued from previous page)

```
        "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:GetVpnConnectionDeviceSampleConfiguration",
        "ec2:GetVpnConnectionDeviceTypes",
        "ec2:GetVpnTunnelReplacementStatus",
        "ec2:ImportClientVpnClientCertificateRevocationList",
        "ec2:ModifyClientVpnEndpoint",
        "ec2:ModifyVpnConnection",
        "ec2:ModifyVpnConnectionOptions",
        "ec2:ModifyVpnTunnelCertificate",
        "ec2:ModifyVpnTunnelOptions",
        "ec2:ReplaceVpnTunnel",
        "ec2:RevokeClientVpnIngress",
        "ec2:TerminateClientVpnConnections"
    ],
    "Resource": "*",
    "Effect": "Deny"
}
```

3.3.3 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

Result

```
new Statement S3  //
   allow
   allListActions
```

```
{
   "Action"  [
       "s3:ListAccessGrants"
       "s3:ListAccessGrantsInstances"
       "s3:ListAccessGrantsLocations"
       "s3:ListAccessPoints"
       "s3:ListAccessPointsForObjectLambda"
       "s3:ListAllMyBuckets"
       "s3:ListBucket"
       "s3:ListBucketMultipartUploads"
       "s3:ListBucketVersions"
       "s3:ListJobs"
       "s3:ListMultiRegionAccessPoints"
       "s3:ListMultipartUploadParts"
       "s3:ListStorageLensConfigurations"
       "s3:ListStorageLensGroups"
       "s3:ListTagsForResource"
   ]
   "Resource"  "*"
   "Effect"  "Allow"
}
```

allReadActions

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

JavaScript

Result

```
new Statement S3  //
   allow
   allReadActions
```

```
{
   "Action"  [
       "s3:DescribeJob"
       "s3:DescribeMultiRegionAccessPointOperation"
       "s3:GetAccelerateConfiguration"
       "s3:GetAccessGrant"
       "s3:GetAccessGrantsInstance"
       "s3:GetAccessGrantsInstanceForPrefix"
```

(continues on next page)

(continued from previous page)

```
        "s3:GetAccessGrantsInstanceResourcePolicy"
        "s3:GetAccessGrantsLocation"
        "s3:GetAccessPoint"
        "s3:GetAccessPointConfigurationForObjectLambda"
        "s3:GetAccessPointForObjectLambda"
        "s3:GetAccessPointPolicy"
        "s3:GetAccessPointPolicyForObjectLambda"
        "s3:GetAccessPointPolicyStatus"
        "s3:GetAccessPointPolicyStatusForObjectLambda"
        "s3:GetAccountPublicAccessBlock"
        "s3:GetAnalyticsConfiguration"
        "s3:GetBucketAcl"
        "s3:GetBucketCORS"
        "s3:GetBucketLocation"
        "s3:GetBucketLogging"
        "s3:GetBucketNotification"
        "s3:GetBucketObjectLockConfiguration"
        "s3:GetBucketOwnershipControls"
        "s3:GetBucketPolicy"
        "s3:GetBucketPolicyStatus"
        "s3:GetBucketPublicAccessBlock"
        "s3:GetBucketRequestPayment"
        "s3:GetBucketTagging"
        "s3:GetBucketVersioning"
        "s3:GetBucketWebsite"
        "s3:GetDataAccess"
        "s3:GetEncryptionConfiguration"
        "s3:GetIntelligentTieringConfiguration"
        "s3:GetInventoryConfiguration"
        "s3:GetJobTagging"
        "s3:GetLifecycleConfiguration"
        "s3:GetMetricsConfiguration"
        "s3:GetMultiRegionAccessPoint"
        "s3:GetMultiRegionAccessPointPolicy"
        "s3:GetMultiRegionAccessPointPolicyStatus"
        "s3:GetMultiRegionAccessPointRoutes"
        "s3:GetObject"
        "s3:GetObjectAcl"
        "s3:GetObjectAttributes"
        "s3:GetObjectLegalHold"
        "s3:GetObjectRetention"
        "s3:GetObjectTagging"
        "s3:GetObjectTorrent"
        "s3:GetObjectVersion"
        "s3:GetObjectVersionAcl"
        "s3:GetObjectVersionAttributes"
        "s3:GetObjectVersionForReplication"
        "s3:GetObjectVersionTagging"
        "s3:GetObjectVersionTorrent"
        "s3:GetReplicationConfiguration"
        "s3:GetStorageLensConfiguration"
        "s3:GetStorageLensConfigurationTagging"
```

(continues on next page)

(continued from previous page)

```
        "s3:GetStorageLensDashboard",
        "s3:GetStorageLensGroup"
    ],
    "Resource": "*",
    "Effect": "Allow"
}
```

allWriteActions

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

JavaScript

Result

```
new Statement.S3()
    .allow()
    .allWriteActions()
```

```
{
    "Action": [
        "s3:AbortMultipartUpload",
        "s3:AssociateAccessGrantsIdentityCenter",
        "s3:CreateAccessGrant",
        "s3:CreateAccessGrantsInstance",
        "s3:CreateAccessGrantsLocation",
        "s3:CreateAccessPoint",
        "s3:CreateAccessPointForObjectLambda",
        "s3:CreateBucket",
        "s3:CreateJob",
        "s3:CreateMultiRegionAccessPoint",
        "s3:CreateStorageLensGroup",
        "s3:DeleteAccessGrant",
        "s3:DeleteAccessGrantsInstance",
        "s3:DeleteAccessGrantsInstanceResourcePolicy",
        "s3:DeleteAccessGrantsLocation",
        "s3:DeleteAccessPoint",
        "s3:DeleteAccessPointForObjectLambda",
        "s3:DeleteBucket",
        "s3:DeleteBucketWebsite",
        "s3:DeleteMultiRegionAccessPoint",
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteStorageLensConfiguration",
        "s3:DeleteStorageLensGroup",
        "s3:DissociateAccessGrantsIdentityCenter",
        "s3:InitiateReplication",
        "s3:PutAccelerateConfiguration",
        "s3:PutAccessGrantsInstanceResourcePolicy",
        "s3:PutAccessPointConfigurationForObjectLambda",
        "s3:PutAnalyticsConfiguration",
        "s3:PutBucketCORS",
```

(continues on next page)

(continued from previous page)

```
        "s3:PutBucketLogging",
        "s3:PutBucketNotification",
        "s3:PutBucketObjectLockConfiguration",
        "s3:PutBucketOwnershipControls",
        "s3:PutBucketRequestPayment",
        "s3:PutBucketVersioning",
        "s3:PutBucketWebsite",
        "s3:PutEncryptionConfiguration",
        "s3:PutIntelligentTieringConfiguration",
        "s3:PutInventoryConfiguration",
        "s3:PutLifecycleConfiguration",
        "s3:PutMetricsConfiguration",
        "s3:PutObject",
        "s3:PutObjectLegalHold",
        "s3:PutObjectRetention",
        "s3:PutReplicationConfiguration",
        "s3:PutStorageLensConfiguration",
        "s3:ReplicateDelete",
        "s3:ReplicateObject",
        "s3:RestoreObject",
        "s3:SubmitMultiRegionAccessPointRoutes",
        "s3:UpdateAccessGrantsLocation",
        "s3:UpdateJobPriority",
        "s3:UpdateJobStatus",
        "s3:UpdateStorageLensGroup"
    ],
    "Resource": "*",
    "Effect": "Allow"
}
```

allPermissionManagementActions

Adds all actions with access level **permission management** to the statement.

JavaScript

Result

```
new Statement.S3()
    .allow()
    .allPermissionManagementActions()
```

```
{
    "Action": [
        "s3:BypassGovernanceRetention",
        "s3:DeleteAccessPointPolicy",
        "s3:DeleteAccessPointPolicyForObjectLambda",
        "s3:DeleteBucketPolicy",
        "s3:ObjectOwnerOverrideToBucketOwner",
        "s3:PutAccessPointPolicy",
        "s3:PutAccessPointPolicyForObjectLambda",
        "s3:PutAccessPointPublicAccessBlock",
```

(continues on next page)

(continued from previous page)

```
        "s3:PutAccountPublicAccessBlock",
        "s3:PutBucketAcl",
        "s3:PutBucketPolicy",
        "s3:PutBucketPublicAccessBlock",
        "s3:PutMultiRegionAccessPointPolicy",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
    ],
    "Resource": "*",
    "Effect": "Allow"
}
```

allTaggingActions

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

JavaScript

Result

```
new Statement S3(  //
   allow()
   allTaggingActions())
```

```
{
    "Action": [
        "s3:DeleteJobTagging",
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersionTagging",
        "s3:DeleteStorageLensConfigurationTagging",
        "s3:PutBucketTagging",
        "s3:PutJobTagging",
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging",
        "s3:PutStorageLensConfigurationTagging",
        "s3:ReplicateTags",
        "s3:TagResource",
        "s3:UntagResource"
    ],
    "Resource": "*",
    "Effect": "Allow"
}
```

3.4 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

Result

```
new Statement.Ec2()
   .allow()
   .toStartInstances()
   .ifEncrypted()
   .ifInstanceType('t3.micro', 't3.nano')
   .ifAssociatePublicIpAddress(false)
   .ifAwsRequestTag('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.

Note: Operators can be passed as string, though it is recommended to use the Operators provided by the package.

JavaScript

Result

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

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

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

JavaScript

Result

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

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

3.5 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

Result

```
new Statement.S3()
   .allow()
   .allActions()
   .onBucket('example-bucket')
   .onObject('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

Result

```
new Statement.S3({ //
  allow(),
  allActions(),
  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"
}
```

Non-global resource ARNs contain the region and/or account. Generally all ARNs contain the partition. In `cdk-iam-floyd` the *account*, *region* and *partition* default to the values provided by the stack. In `iam-floyd` the *partition* defaults to `aws` and the *account* and *region* default to `*`.

The `on*()` methods take optional parameters to override the default values:

JavaScript

Result

```
new Statement.Lambda({
  allow(),
  toUpdateFunctionCode(),
  onFunction('my-function', '098765432109', 'us-east-1', 'aws'),
});
```

```
{
    "Action": "lambda:UpdateFunctionCode",
    "Resource": "arn:aws:lambda:us-east-1:098765432109:function:my-function",
    "Effect": "Allow"
}
```

If you want to override the defaults for the whole statement, see *in (ARN defaults)*.

3.6 in (ARN defaults)

The *on** methods generate ARNs which contain *partition* and potentially *region* and *account*. The `in()*` methods can be used to override the defaults for all **consecutively** added resources. You allow/deny something on resources **in** a specific account, region and partition.

Note: The `in*()` methods do not by themselves modify the statement. They just set the defaults for the resource added **consecutively** to the statement. Therefore make sure to call the `in*()` methods before adding resources via `on*()`.

JavaScript

Result

```
new Statement.Lambda()
    .allow()
    .toUpdateFunctionCode()
    .inAccount('098765432109')
    .inRegion('us-east-1')
    .inPartition('aws')
    .onFunction('my-function-1')
    .onFunction('my-function-2')
```

```
{
  "Action": "lambda:UpdateFunctionCode",
  "Resource": [
      "arn:aws:lambda:us-east-1:098765432109:function:my-function-1",
      "arn:aws:lambda:us-east-1:098765432109:function:my-function-2"
  ],
  "Effect": "Allow"
}
```

There also is a shorthand function to set all defaults at once:

JavaScript

Result

```
new Statement.Lambda()
    .allow()
    .toUpdateFunctionCode()
    .in('098765432109', 'us-west-1', 'aws')
    .onFunction('my-function-1')
    .onFunction('my-function-2')
```

```
{
  "Action": "lambda:UpdateFunctionCode",
  "Resource": [
      "arn:aws:lambda:us-west-1:098765432109:function:my-function-1",
      "arn:aws:lambda:us-west-1:098765432109:function:my-function-2"
  ],
  "Effect": "Allow"
}
```

Since these methods set defaults for consecutively added resources, you can also override the defaults for additional resource in the same statement:

JavaScript

```
new Statement.Lambda()
    .allow()
    .toUpdateFunctionCode()
    .in('098765432109', 'us-west-1', 'aws')
    .onFunction('my-function-1')
    .in('123456789012', 'us-east-1', 'aws')
    .onFunction('my-function-2')
```

Result

```
{
  "Action": "lambda:UpdateFunctionCode",
  "Resource": [
      "arn:aws:lambda:us-west-1:098765432109:function:my-function-1",
      "arn:aws:lambda:us-east-1:123456789012:function:my-function-2"
  ],
  "Effect": "Allow"
}
```

3.7 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

```
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()
```

(continues on next page)

(continued from previous page)

```
  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');

const s12 = new Statement.Sts()
  allow()
  toAssumeRole()
  forCanonicalUser('userID');

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

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

(continues on next page)

(continued from previous page)

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

(continues on next page)

(continued from previous page)

```
    "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",
    "Principal": {
        "AWS": [
            "arn:aws:sts::123456789:assumed-role/role-name/session-name"
        ]
    }
},
{
    "Action": "sts:AssumeRole",
    "Effect": "Allow",
    "Principal": {
        "CanonicalUser": [
            "userID"
        ]
    }
```

(continues on next page)

(continued from previous page)

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

Some of the `for*` methods accept multiple values at once:

JavaScript

Result

```
const s1 = new Statement.Sts()
    .allow()
    .toAssumeRole()
    .forAccount('1234567890', '0987654321');

// when you already have a list:
const accounts = ['1234567890', '0987654321'];
const s2 = new Statement.Sts()
    .allow()
    .toAssumeRole()
    .forAccount(...accounts);

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

// when you already have a list:
const users = ['Bob', 'John'];
const s4 = new Statement.Sts()
    .allow()
    .toAssumeRole()
    .forUser('1234567890', ...users);
```

```
{
    "Action": "sts:AssumeRole",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::1234567890:root",
            "arn:aws:iam::0987654321:root"
        ]
    }
}
{
    "Action": "sts:AssumeRole",
```

(continues on next page)

(continued from previous page)

```
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::1234567890:root",
            "arn:aws:iam::0987654321:root"
        ]
    }
},
{
    "Action": "sts:AssumeRole",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::1234567890:user/Bob",
            "arn:aws:iam::1234567890:user/John"
        ]
    }
},
{
    "Action": "sts:AssumeRole",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::1234567890:user/Bob",
            "arn:aws:iam::1234567890:user/John"
        ]
    }
}
```

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

JavaScript

Result

```
new Statement.Sts()
   .allow()
   .toAssumeRole()
   .forCdkPrincipal(
    new iam.ServicePrincipal('sns.amazonaws.com'),
    new 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)

```
}
```

3.8 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.

3.8.1 notAction

Switches the policy provider to use NotAction.

JavaScript

Result

```
new Statement.S3()
  .allow()
  .notAction()
  .toDeleteBucket()
  .onBucket('example-bucket')
```

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

3.8.2 notResource

Switches the policy provider to use NotResource.

JavaScript

Result

```
new Statement.S3()
  .allow()
  .notResource()
  .toDeleteBucket()
  .onBucket('example-bucket')
```

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

3.8.3 notPrincipal

Switches the policy provider to use NotPrincipal.

JavaScript

Result

```
new Statement.S3()
   .deny()
   .allActions()
   .notPrincipal()
   .forUser('1234567890', 'Bob')
   .onObject('example-bucket', '*')
```

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

3.9 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

Result

```
new Statement.Ec2(//
   .allow()
   .allReadActions()
   .allListActions()
   .compact()
```

```
{
    "Action": [
        "ec2:Describe*",
        "ec2:ExportClientVpnClientC*",
        "ec2:Get*",
        "ec2:List*",
        "ec2:SearchLocalGatewayRoutes",
        "ec2:SearchTransitGateway*"
```

(continues on next page)

(continued from previous page)

```
    "Resource"  "*"
    "Effect"  "Allow"
}
```

CHAPTER

FOUR

OPERATORS

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.

Condition operators are available though the `Operator` class. First import it along with the `Statement`:

TypeScript

JavaScript

```
// for use without AWS CDK use the iam-floyd package
import { Operator, Statement } from 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
import { Operator, Statement } from 'cdk-iam-floyd'
```

```
// for use without AWS CDK use the iam-floyd package
const { Operator, Statement } = require 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
const { Operator, Statement } = require 'cdk-iam-floyd'
```

Operators can be simple strings such as `StringEquals` or get complex with modifiers such as `ForAnyValue` or `IfExists`. For simple operators you can use the static properties of the `Operator` class:

JavaScript

Result

```
new Statement.Ec2()
  .allow()
  .toStartInstances()
  .ifAwsRequestTag('TagWithSpecialChars', '*John*', Operator.stringEquals)
```

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

(continues on next page)

(continued from previous page)

```
    "Resource": "*",
    "Effect": "Allow"
}
```

Complex operators can be generated by instantiating the `Operator` class and calling its methods:

JavaScript

Result

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

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

JavaScript

Result

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

```
{
    "Condition": {
        "ForAnyValue:StringEquals": {
            "dynamodb:Attributes": [
                "ID",
                "PostDateTime"
```

(continues on next page)

(continued from previous page)

```
            ]
        }
    },
    "Action": "dynamodb:PutItem",
    "Resource": "arn:aws:dynamodb:*:*:table/Thread",
    "Effect": "Deny"
}
```

JavaScript

Result

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

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

CHAPTER

FIVE

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

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
  ]
}
```

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

(continues on next page)

(continued from previous page)

```
            "Condition": {
                "StringLike": {
                    "ec2:ResourceTag/Owner": "${aws:username}"
                }
            },
            "Action": "ec2:StopInstances",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddressTransfers",
                "ec2:DescribeAddresses",
                "ec2:DescribeAddressesAttribute",
                "ec2:DescribeAggregateIdFormat",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeAwsNetworkPerformanceMetricSubscriptions",
                "ec2:DescribeBundleTasks",
                "ec2:DescribeByoipCidrs",
                "ec2:DescribeCapacityBlockOfferings",
                "ec2:DescribeCapacityReservationFleets",
                "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:DescribeFastLaunchImages",
                "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",
```

(continues on next page)

(continued from previous page)

```
                "ec2:DescribeImageAttribute"
                "ec2:DescribeImages"
                "ec2:DescribeImportImageTasks"
                "ec2:DescribeImportSnapshotTasks"
                "ec2:DescribeInstanceAttribute"
                "ec2:DescribeInstanceConnectEndpoints"
                "ec2:DescribeInstanceCreditSpecifications"
                "ec2:DescribeInstanceEventNotificationAttributes"
                "ec2:DescribeInstanceEventWindows"
                "ec2:DescribeInstanceStatus"
                "ec2:DescribeInstanceTopology"
                "ec2:DescribeInstanceTypeOfferings"
                "ec2:DescribeInstanceTypes"
                "ec2:DescribeInstances"
                "ec2:DescribeInternetGateways"
                "ec2:DescribeIpamByoasn"
                "ec2:DescribeIpamPools"
                "ec2:DescribeIpamResourceDiscoveries"
                "ec2:DescribeIpamResourceDiscoveryAssociations"
                "ec2:DescribeIpamScopes"
                "ec2:DescribeIpams"
                "ec2:DescribeIpv6Pools"
                "ec2:DescribeKeyPairs"
                "ec2:DescribeLaunchTemplateVersions"
                "ec2:DescribeLaunchTemplates"
                "ec2:DescribeLocalGatewayRouteTablePermissions"
                "ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations"
                "ec2:DescribeLocalGatewayRouteTableVpcAssociations"
                "ec2:DescribeLocalGatewayRouteTables"
                "ec2:DescribeLocalGatewayVirtualInterfaceGroups"
                "ec2:DescribeLocalGatewayVirtualInterfaces"
                "ec2:DescribeLocalGateways"
                "ec2:DescribeLockedSnapshots"
                "ec2:DescribeMacHosts"
                "ec2:DescribeManagedPrefixLists"
                "ec2:DescribeMovingAddresses"
                "ec2:DescribeNatGateways"
                "ec2:DescribeNetworkAcls"
                "ec2:DescribeNetworkInsightsAccessScopeAnalyses"
                "ec2:DescribeNetworkInsightsAccessScopes"
                "ec2:DescribeNetworkInsightsAnalyses"
                "ec2:DescribeNetworkInsightsPaths"
                "ec2:DescribeNetworkInterfaceAttribute"
                "ec2:DescribeNetworkInterfacePermissions"
                "ec2:DescribeNetworkInterfaces"
                "ec2:DescribePlacementGroups"
                "ec2:DescribePrefixLists"
                "ec2:DescribePrincipalIdFormat"
                "ec2:DescribePublicIpv4Pools"
                "ec2:DescribeRegions"
                "ec2:DescribeReplaceRootVolumeTasks"
                "ec2:DescribeReservedInstances"
```

(continues on next page)

(continued from previous page)

```
                "ec2:DescribeReservedInstancesListings"
                "ec2:DescribeReservedInstancesModifications"
                "ec2:DescribeReservedInstancesOfferings"
                "ec2:DescribeRouteTables"
                "ec2:DescribeScheduledInstanceAvailability"
                "ec2:DescribeScheduledInstances"
                "ec2:DescribeSecurityGroupReferences"
                "ec2:DescribeSecurityGroupRules"
                "ec2:DescribeSecurityGroups"
                "ec2:DescribeSnapshotAttribute"
                "ec2:DescribeSnapshotTierStatus"
                "ec2:DescribeSnapshots"
                "ec2:DescribeSpotDatafeedSubscription"
                "ec2:DescribeSpotFleetInstances"
                "ec2:DescribeSpotFleetRequestHistory"
                "ec2:DescribeSpotFleetRequests"
                "ec2:DescribeSpotInstanceRequests"
                "ec2:DescribeSpotPriceHistory"
                "ec2:DescribeStaleSecurityGroups"
                "ec2:DescribeStoreImageTasks"
                "ec2:DescribeSubnets"
                "ec2:DescribeTags"
                "ec2:DescribeTrafficMirrorFilters"
                "ec2:DescribeTrafficMirrorSessions"
                "ec2:DescribeTrafficMirrorTargets"
                "ec2:DescribeTransitGatewayAttachments"
                "ec2:DescribeTransitGatewayConnectPeers"
                "ec2:DescribeTransitGatewayConnects"
                "ec2:DescribeTransitGatewayMulticastDomains"
                "ec2:DescribeTransitGatewayPeeringAttachments"
                "ec2:DescribeTransitGatewayPolicyTables"
                "ec2:DescribeTransitGatewayRouteTableAnnouncements"
                "ec2:DescribeTransitGatewayRouteTables"
                "ec2:DescribeTransitGatewayVpcAttachments"
                "ec2:DescribeTransitGateways"
                "ec2:DescribeTrunkInterfaceAssociations"
                "ec2:DescribeVerifiedAccessEndpoints"
                "ec2:DescribeVerifiedAccessGroups"
                "ec2:DescribeVerifiedAccessInstanceLoggingConfigurations"
                "ec2:DescribeVerifiedAccessInstanceWebAclAssociations"
                "ec2:DescribeVerifiedAccessInstances"
                "ec2:DescribeVerifiedAccessTrustProviders"
                "ec2:DescribeVolumeAttribute"
                "ec2:DescribeVolumeStatus"
                "ec2:DescribeVolumes"
                "ec2:DescribeVolumesModifications"
                "ec2:DescribeVpcAttribute"
                "ec2:DescribeVpcClassicLink"
                "ec2:DescribeVpcClassicLinkDnsSupport"
                "ec2:DescribeVpcEndpointConnectionNotifications"
                "ec2:DescribeVpcEndpointConnections"
                "ec2:DescribeVpcEndpointServiceConfigurations"
```

(continues on next page)

(continued from previous page)

```
                "ec2:DescribeVpcEndpointServicePermissions"
                "ec2:DescribeVpcEndpointServices"
                "ec2:DescribeVpcEndpoints"
                "ec2:DescribeVpcPeeringConnections"
                "ec2:DescribeVpcs"
                "ec2:DescribeVpnConnections"
                "ec2:DescribeVpnGateways"
                "ec2:ExportClientVpnClientCertificateRevocationList"
                "ec2:ExportClientVpnClientConfiguration"
                "ec2:GetAssociatedEnclaveCertificateIamRoles"
                "ec2:GetAssociatedIpv6PoolCidrs"
                "ec2:GetAwsNetworkPerformanceData"
                "ec2:GetCapacityReservationUsage"
                "ec2:GetCoipPoolUsage"
                "ec2:GetConsoleOutput"
                "ec2:GetConsoleScreenshot"
                "ec2:GetDefaultCreditSpecification"
                "ec2:GetEbsDefaultKmsKeyId"
                "ec2:GetEbsEncryptionByDefault"
                "ec2:GetFlowLogsIntegrationTemplate"
                "ec2:GetGroupsForCapacityReservation"
                "ec2:GetHostReservationPurchasePreview"
                "ec2:GetImageBlockPublicAccessState"
                "ec2:GetInstanceMetadataDefaults"
                "ec2:GetInstanceTypesFromInstanceRequirements"
                "ec2:GetInstanceUefiData"
                "ec2:GetIpamAddressHistory"
                "ec2:GetIpamDiscoveredAccounts"
                "ec2:GetIpamDiscoveredPublicAddresses"
                "ec2:GetIpamDiscoveredResourceCidrs"
                "ec2:GetIpamPoolAllocations"
                "ec2:GetIpamPoolCidrs"
                "ec2:GetIpamResourceCidrs"
                "ec2:GetLaunchTemplateData"
                "ec2:GetManagedPrefixListAssociations"
                "ec2:GetManagedPrefixListEntries"
                "ec2:GetNetworkInsightsAccessScopeAnalysisFindings"
                "ec2:GetNetworkInsightsAccessScopeContent"
                "ec2:GetPasswordData"
                "ec2:GetReservedInstancesExchangeQuote"
                "ec2:GetResourcePolicy"
                "ec2:GetSecurityGroupsForVpc"
                "ec2:GetSerialConsoleAccessStatus"
                "ec2:GetSnapshotBlockPublicAccessState"
                "ec2:GetSpotPlacementScores"
                "ec2:GetSubnetCidrReservations"
                "ec2:GetTransitGatewayAttachmentPropagations"
                "ec2:GetTransitGatewayMulticastDomainAssociations"
                "ec2:GetTransitGatewayPolicyTableAssociations"
                "ec2:GetTransitGatewayPolicyTableEntries"
                "ec2:GetTransitGatewayPrefixListReferences"
                "ec2:GetTransitGatewayRouteTableAssociations"
```

(continues on next page)

(continued from previous page)

```
                "ec2:GetTransitGatewayRouteTablePropagations"
                "ec2:GetVerifiedAccessEndpointPolicy"
                "ec2:GetVerifiedAccessGroupPolicy"
                "ec2:GetVerifiedAccessInstanceWebAcl"
                "ec2:GetVpnConnectionDeviceSampleConfiguration"
                "ec2:GetVpnConnectionDeviceTypes"
                "ec2:GetVpnTunnelReplacementStatus"
                "ec2:ListImagesInRecycleBin"
                "ec2:ListSnapshotsInRecycleBin"
                "ec2:SearchLocalGatewayRoutes"
                "ec2:SearchTransitGatewayMulticastGroups"
                "ec2:SearchTransitGatewayRoutes"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

JavaScript

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(),
  ],
};
```

```
{
    "Version": "2012-10-17",
```

(continues on next page)

(continued from previous page)

```
    "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:CloseAccount",
                "account:DeleteAlternateContact",
                "account:DisableRegion",
                "account:EnableRegion",
                "account:PutAlternateContact",
                "account:PutChallengeQuestions",
                "account:PutContactInformation"
            ],
            "Resource": "*",
            "Effect": "Deny"
        },
        {
            "Action": [
                "organizations:AcceptHandshake",
                "organizations:AttachPolicy",
                "organizations:CancelHandshake",
                "organizations:CloseAccount",
                "organizations:CreateAccount",
                "organizations:CreateGovCloudAccount",
                "organizations:CreateOrganization",
                "organizations:CreateOrganizationalUnit",
                "organizations:CreatePolicy",
                "organizations:DeclineHandshake",
                "organizations:DeleteOrganization",
                "organizations:DeleteOrganizationalUnit",
                "organizations:DeletePolicy",
                "organizations:DeleteResourcePolicy",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:DetachPolicy",
```

(continues on next page)

(continued from previous page)

```
                "organizations:DisableAWSServiceAccess"
                "organizations:DisablePolicyType"
                "organizations:EnableAWSServiceAccess"
                "organizations:EnableAllFeatures"
                "organizations:EnablePolicyType"
                "organizations:InviteAccountToOrganization"
                "organizations:LeaveOrganization"
                "organizations:MoveAccount"
                "organizations:PutResourcePolicy"
                "organizations:RegisterDelegatedAdministrator"
                "organizations:RemoveAccountFromOrganization"
                "organizations:UpdateOrganizationalUnit"
                "organizations:UpdatePolicy"
            ],
            "Resource": "*"
            "Effect": "Deny"
        }
    ]
}
```

CHAPTER

SIX

COLLECTIONS

Note: The list of collections is not exhaustive. If you have a list of statements that you think is worth sharing with others, please open an issue or a pull request.

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.

First import the `Collection` provider:

TypeScript

JavaScript

```
// for use without AWS CDK use the iam-floyd package
import { Collection } from 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
import { Collection } from 'cdk-iam-floyd'
```

```
// for use without AWS CDK use the iam-floyd package
const { Collection } = require 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
const { Collection } = require 'cdk-iam-floyd'
```

Collections then can be called via:

JavaScript

Result

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

```
{
    "Condition": {
        "StringLike": {
            "aws:RequestTag/Owner": "${aws:username}"
        }
```

(continues on next page)

(continued from previous page)

```
    },
    "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

Result

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

```
{
    "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"
        }
    ]
}
```

6.1 Available collections

6.1.1 allowEc2InstanceDeleteByOwner

Allows stopping EC2 instance for the user who started them.

CHAPTER

SEVEN

AWS MANAGED POLICIES

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 `AwsManagedPolicy` class provides an up-to-date collection of AWS managed policies. This helps adding managed policies to IAM roles and users in a type-safe way.

The class provides the **names** of the policies. If you instead need the ARN, prefix the string with `arn:aws:iam::aws:policy/`.

The package `cdk-iam-floyd` additionally provides methods for directly creating `aws_iam.IManagedPolicy` objects.

First import `AwsManagedPolicy`:

TypeScript

JavaScript

```
// for use without AWS CDK use the iam-floyd package
import { AwsManagedPolicy } from 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
import { AwsManagedPolicy } from 'cdk-iam-floyd'
```

```
// for use without AWS CDK use the iam-floyd package
const { AwsManagedPolicy } = require 'iam-floyd'

// for use with CDK use the cdk-iam-floyd package
const { AwsManagedPolicy } = require 'cdk-iam-floyd'
```

Usage in aws-sdk v3 and aws-cdk:

aws-cdk

aws-sdk

```
readOnlyRole.addManagedPolicy(
  new AwsManagedPolicy('ReadOnlyAccess').
);
```

```
await iamClient.send(
  new AttachRolePolicyCommand({
    RoleName: 'ReadOnlyRole',
```

(continues on next page)

(continued from previous page)

```
    PolicyArn: `arn:aws:iam::aws:policy/${AwsManagedPolicy ReadOnlyAccess}`
  }),
);
```

CHAPTER

EIGHT

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.

8.1 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

Result

```
new Statement S3()
  .allow()
  .allActions()
  .onBucket('example-bucket')
  .onObject('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.

8.2 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, please report it through the feedback functionality on the IAM documentation page of the related service. Shortly after changes have been added to the documentation, they will be available in this package.

8.3 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.

8.4 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 `^2.0.0` and so far I have not seen any issues.

8.5 Is the package following semantic versioning?

> **Warning:** The package has not reached a stable state yet. Therefore breaking changes are not yet reflected by a major update!

When the package has reached version 1, manual changes by developers of this package follow 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.

8.6 How can I set a statement SID?

The SID can be passed as parameter to the constructor:

JavaScript

Result

```
new Statement Ec2 'MYSID' //
  allow()
  toStartInstances()
  toStopInstances()
```

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

8.7 I don't like method chaining!

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

JavaScript

Result

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

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

8.8 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.

CHAPTER

NINE

LEGAL

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.

9.1 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.

CHAPTER

TEN

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:

- 393 Services
- 16657 Actions
- 1787 Resource Types
- 1738 Condition keys

10.1 Similar projects

- cdk-iam-actions
- cdk-iam-generator
- cdk-iam-policy-builder-helper
- iam-policy-generator
- policyuniverse
- policy_sentry