AWS IAM Roles and Permissions

AWS IAM Roles and Permissions

Note Additional information on AWS IAM roles and permissions is available in the *Cisco Cloud Network Controller AWS User Guide,* including how to configure an AWS provider as one of the following types of tenants:

- Trusted tenant
- Untrusted tenant
- Organization tenant, supported in Release 4.2(3) and later

Specific AWS IAM roles and permissions are required for the installation and operation of the Cisco Cloud Network Controller.

```
{
 "Version": "2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "iam:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "ec2:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "cloudformation:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "s3:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "sns:*",
   "Resource": "*"
  }
 ]
}
```

When installing Cisco Cloud Network Controller using the CloudFormation template (CFT), we recommend installation by a user who has the full Administrator Access on AWS (for example, by a user who has the permission policy ARN **arn:aws:iam::aws:policy/AdministratorAccess** attached to it, either directly, by using a role policy, or with a user group). However, if there is no user with AWS Administrator Access available, the user installing Cisco Cloud Network Controller must have this minimum set of permissions:

The above permission set is necessary for a user who installs Cisco Cloud Network Controller using the CFT. Following are more detailed descriptions of each of the required permissions presented above, as shown in the **Action** lines:

- **iam Permissions:** The Cisco Cloud Network Controller instance is an AWS EC2 instance that runs with an AWS role called **ApicAdmin**. This role needs to be created by the CloudFormation stack. Running the Cisco Cloud Network Controller instance with the **ApicAdmin** role allows the Cisco Cloud Network Controller instance to get temporary credentials using the AWS metadata service. This frees the Cisco Cloud Network Controller instance from having to use fixed access key IDs and secret access keys for making AWS API calls.

- **ec2 Permissions:** Needed so that the stack can create the needed VPC, subnets, security groups, and so on. The stack creates the infra VPC, where the Cisco Cloud Network Controller instance is deployed.

- **cloudformation Permissions:** Needed to run the CFT itself.

- **s3 Permissions:** Needed so that the CFT is saved in an S3 bucket based on the needs of the AWS CloudFormation stack.

- **sns Permissions:** Needed to get notifications for running the CloudFormation stack.

For operations, Cisco Cloud Network Controller runs with **ApicAdmin** role. This role has two policies attached, and they get created as part of launching the CloudFormation template:

- **ApicAdminFullAccess Policy:** Permissions listed in this policy allows Cisco Cloud Network Controller to create and manage EC2 and VPC resources, S3 buckets, Resource Groups, account notifications and logs. Note that Cisco Cloud Network Controller only tries to manage the resources it creates. It does not deal with resources created by any other applications.

  This policy should have the following permissions:

```
{
 "Version": "2012-10-17",
 "Statement": ["Resources": {
                                "rApicAdminFullAccessPolicy": {
                                                "Type": "AWS::IAM::ManagedPolicy",
                                                "Properties": {
                                                                "Description": "Full
Access for ApicAdmin Role",
                                                                "ManagedPolicyName":
"ApicAdminFullAccess",
                                                                "Path": "/",
                                                                "PolicyDocument": {
                                                                            "Version":
 "2012-10-17",

"Statement": [{
```

```
"Effect": "Allow",

"Action": "organizations:*",

"Resource": "*"
          }, {
              "Effect": "Allow",
              "Action": [
                  "ec2:DeleteCustomerGateway",
                  "ec2:DeleteInternetGateway",
                  "ec2:DeleteKeyPair",
                  "ec2:DeleteSecurityGroup",
                  "ec2:DeleteSubnet",
                  "ec2:DeleteTransitGateway*",
                  "ec2:DeleteVpc*",
                  "ec2:DeleteVpn*"
              ],
              "Resource": "*",
              "Condition": {
                  "StringEquals": {
                      "ec2:ResourceTag/CiscoAciCapic": ""
                  }
              }
          },{
              "Effect": "Allow",
              "Action": [
                  "ec2:AcceptTransitGatewayPeeringAttachment",
                  "ec2:AcceptVpcPeeringConnection",
                  "ec2:AllocateAddress",
                  "ec2:AssignPrivateIpAddresses",
                  "ec2:AssociateAddress",
                  "ec2:AssociateRouteTable",
                  "ec2:AssociateTransitGatewayRouteTable",
                  "ec2:AssociateVpcCidrBlock",
                  "ec2:AttachInternetGateway",
                  "ec2:AttachNetworkInterface",
                  "ec2:AttachVpnGateway",
                  "ec2:AuthorizeSecurityGroup*",
                  "ec2:CreateCustomerGateway",
                  "ec2:CreateFlowLogs",
                  "ec2:CreateInternetGateway",
                  "ec2:CreateKeyPair",
                  "ec2:CreateNetwork*",
                  "ec2:CreateRoute*",
                  "ec2:CreateSecurityGroup",
                  "ec2:CreateSubnet",
                  "ec2:CreateTags",
                  "ec2:CreateTransitGateway*",
                  "ec2:CreateVpc*",
                  "ec2:CreateVpn*",
                  "ec2:DeleteFlowLogs",
                  "ec2:DeleteNetwork*",
                  "ec2:DeleteRoute*",
                  "ec2:DeleteTags",
                  "ec2:DetachInternetGateway",
                  "ec2:DetachNetworkInterface",
                  "ec2:DetachVpnGateway",
                  "ec2:DisableTransitGatewayRouteTablePropagation",
                  "ec2:DisassociateAddress",
                  "ec2:DisassociateRouteTable",
                  "ec2:DisassociateTransitGatewayRouteTable",
                  "ec2:DisassociateVpcCidrBlock",
                  "ec2:EnableTransitGatewayRouteTablePropagation",
            "ec2:EnableVgwRoutePropagation",
            "ec2:GetManagedPrefixListEntries",
            "ec2:GetTransitGatewayRouteTableAssociations",
            "ec2:ModifyInstanceAttribute",
            "ec2:ModifyNetworkInterfaceAttribute",
            "ec2:ModifySubnetAttribute",
            "ec2:ModifyTransitGateway",
            "ec2:ModifyTransitGatewayVpcAttachment",
            "ec2:ModifyVpcAttribute",
            "ec2:ModifyVpcEndpoint",
            "ec2:ReleaseAddress",
            "ec2:ReplaceNetworkAclAssociation",
            "ec2:RevokeSecurityGroup*",
            "ec2:RunInstances",
            "ec2:SearchTransitGatewayRoutes",
            "ec2:StartInstances",
            "ec2:TerminateInstances",
            "ec2:UnassignPrivateIpAddresses"
        ],
        "Resource": "*"
    }, {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "*"
    }, {
        "Effect": "Allow",
        "Action": [
            "sqs:CreateQueue",
            "sqs:DeleteMessage",
            "sqs:DeleteQueue",
            "sqs:GetQueueAttributes",
            "sqs:GetQueueUrl",
            "sqs:ListQueueTags",
            "sqs:ListQueues",
            "sqs:ReceiveMessage",
            "sqs:SetQueueAttributes",
            "sqs:TagQueue"
        ],
        "Resource": "*"
    }, {
        "Effect": "Allow",
        "Action": [
            "elasticloadbalancing:AddListenerCertificates",
            "elasticloadbalancing:AddTags",
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:CreateLoadBalancer",
            "elasticloadbalancing:CreateRule",
            "elasticloadbalancing:CreateTargetGroup",
            "elasticloadbalancing:DeleteListener",
            "elasticloadbalancing:DeleteLoadBalancer",
            "elasticloadbalancing:DeleteRule",
            "elasticloadbalancing:DeleteTargetGroup",
            "elasticloadbalancing:DeregisterTargets",
            "elasticloadbalancing:ModifyListener",
            "elasticloadbalancing:ModifyRule",
            "elasticloadbalancing:ModifyTargetGroup",
            "elasticloadbalancing:RegisterTargets",
            "elasticloadbalancing:RemoveListenerCertificates",
            "elasticloadbalancing:RemoveTags",
            "elasticloadbalancing:SetIpAddressType",
            "elasticloadbalancing:SetRulePriorities",
            "elasticloadbalancing:SetSecurityGroups",
            "elasticloadbalancing:SetSubnets"
        ],
            "Resource": "*"
        }, {
            "Effect": "Allow",
            "Action": [
                "acm:DeleteCertificate",
                "acm:ImportCertificate"
            ],
            "Resource": "*"
        }, {
            "Effect": "Allow",
            "Action": ["config:*"],
            "Resource": "*"
        }, {
            "Effect": "Allow",
            "Action": [
                "cloudtrail:AddTags",
                "cloudtrail:CreateTrail",
                "cloudtrail:GetTrailStatus",
                "cloudtrail:StartLogging",
                "cloudtrail:DeleteTrail"
            ],
            "Resource": "*"
        }, {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:DeleteAlarms",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:PutMetricAlarm"
            ],
            "Resource": "*"
        }, {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DeleteLogGroup",
                "logs:DeleteLogStream",
                "logs:FilterLogEvents",
                "logs:ListTagsLogGroup",
                "logs:PutRetentionPolicy",
                "logs:PutLogEvents",
                "logs:TagLogGroup"
            ],
            "Resource": "*"
        }, {
            "Effect": "Allow",
            "Action": [
                "resource-groups:CreateGroup",
                "resource-groups:DeleteGroup",
                "resource-groups:GetGroup",
                "resource-groups:GetGroupQuery",
                "resource-groups:UpdateGroupQuery"
            ],
            "Resource": "*"
        }, {
            "Sid": "CloudWatchEventsFullAccess",
            "Effect": "Allow",
            "Action": [
                "events:DeleteRule",
                "events:DisableRule",
                "events:EnableRule",
                "events:ListRuleNamesByTarget",
                "events:ListRules",
                "events:ListTargetsByRule",
                            "events:PutRule",
                            "events:PutTargets",
                            "events:RemoveTargets"
                        ],
                        "Resource": "*"
                    }, {
                        "Effect": "Allow",
                        "Action": [
                            "ram:AcceptResourceShareInvitation",
                            "ram:CreateResourceShare",
                            "ram:DisassociateResourceShare",
                            "ram:DeleteResourceShare",
                            "ram:GetResourceShareInvitations",
                            "ram:GetResourceShares",
                            "ram:TagResource",
                            "ram:UntagResource"
                        ],
                        "Resource": "*"
                    }, {
                        "Effect": "Allow",
                        "Action": "ssm:GetParameters",
                        "Resource": "*"
                    }, {
                        "Effect": "Allow",
                        "Action": "iam:PassRole",
                        "Resource": { "Fn::Join": [ "", [ "arn:", {"Ref":
"AWS::Partition"}, ":iam::", {"Ref": "AWS::AccountId"}, ":role/ApicAdmin" ] ] }
                    }, {
                        "Effect": "Allow",
                        "Action": [
                            "iam:List*",
                            "iam:Get*",
                            "iam:CreateServiceLinkedRole",
                            "iam:DeleteServiceLinkedRole",
                            "iam:GetServiceLinkedRoleDeletionStatus",
                            "iam:AttachRolePolicy",
                            "iam:PutRolePolicy",
                            "iam:UpdateRoleDescription",
                            "iam:UploadServerCertificate",
                            "iam:DeleteServerCertificate",
                            "iam:UpdateRoleDescription"
                        ],
                        "Resource": "*"
                    },{
                        "Action": [
                            "aws-marketplace:MeterUsage"
                        ],
                        "Effect": "Allow",
                        "Resource": "*"
                    },{
                        "Effect": "Allow",
                        "Action": [
                            "ec2:Describe*",
                            "elasticloadbalancing:Describe*",
                            "cloudtrail:Describe*",
                            "logs:Describe*",
                            "events:Describe*"
                        ],
                        "Resource": "*"
                    }]
                                                                 }
                                                 }
    }
```

- **ApicTenantsAccess Policy:** Permissions listed in this policy allows Cisco Cloud Network Controller to assume the role of tenant accounts and call AWS APIs on those tenant AWS accounts. This allows Cisco Cloud Network Controller to access tenant accounts without having to use the hard credentials of those tenant accounts.

  This policy should have the following permissions:

  ```
  {
   "Version": "2012-10-17",
   "Statement": [{
    "Action": "sts:AssumeRole",
    "Resource": "*",
    "Effect": "Allow"
   }]
  }
  ```

Note that Cisco Cloud Network Controller itself does not need IAM permissions for its operation because it does not create any IAM policies or roles after its installation.

Cisco Cloud Network Controller will attempt to manage the AWS resources that are created by it, but it will not attempt to manage resources created by other applications, other than listing existing resources as inventory. At the same time, AWS IAM users in those accounts (both the infra account and other tenant accounts) should not interfere with the resources created by Cisco Cloud Network Controller. Therefore, all resources created by Cisco Cloud Network Controller on AWS have at least one of these two tags applied on them:

- **AciDnTag**
- **AciOwnerTag**

Therefore, when you create AWS IAM users who have permission to create, delete or update EC2, VPC and other resources, you must prevent these users from accessing or modifying the resources created and managed by Cisco Cloud Network Controller. Such restrictions should apply on both the infra and other user tenant accounts. AWS account administrators should use the above two tags to prevent users from accessing or modifying the resources created and managed by Cisco Cloud Network Controller.

For example, you might have an access policy similar to the following for an IAM user to prevent unintended access to resources managed by Cisco Cloud Network Controller:

```
{
 "Effect": "Deny",
 "Action": [
  "ec2:*"
 ],
 "Resource": "*",
 "Condition": {
  "StringLike": {
   "ec2:ResourceTag/AciDnTag": "*"
  }
 }
}
```