

Implementation Guide

Dynamic Object and Rule Extensions for AWS Network Firewall



Dynamic Object and Rule Extensions for AWS Network Firewall: Implementation Guide

Table of Contents

Solution overview	1
Cost	2
Architecture overview	4
Solution components	6
Amazon DynamoDB	6
Amazon EventBridge	6
AWS Config	6
AWS Network Firewall	6
Amazon Simple Storage Service	6
Amazon Elastic Container Service	6
AWS Lambda	7
Security	8
IAM roles	8
Network configuration	8
Security groups	9
Data protection	9
Domain concepts	10
Roles	10
Rule bundle	10
Rule	10
Object	10
Audit	11
Design considerations	12
Event-driven vs. continuous polling	12
Regional deployments	12
AWS CloudFormation template	13
Deployment	14
Prerequisites	14
Install tools	14
Activate AWS Config and AWS Network Firewall	14
Deployment overview	15
Step 1. Update solution configuration	15
Override default configuration values	17
Step 2. Build and deploy the stack	17

(Optional) Activate cross-account access	18
(Optional) Deploy to an existing VPC	19
Interaction steps	22
Identify a host machine with which to interact with the solution	22
Install the prerequisite tools	23
Create a Network Firewall rule group	23
Use CLI tool to create solution rule bundle entity	23
Create an object to represent the underlying cloud resource	25
Create a rule in the rule bundle	26
Troubleshooting	28
Contact Support	28
Create case	28
How can we help?	28
Additional information	28
Help us resolve your case faster	29
Solve now or contact us	29
Uninstall the solution	30
Using the AWS Management Console	30
Using the AWS Command Line Interface (CLI)	30
Additional resources	31
AWS services	31
Operation and customization	32
Data backup and restore	32
AWS resources and service limitations	32
API schema	33
Install JQ	33
Install AWS curl	33
Install GNU Getopt	34
Customize OPA policy	34
Create an EC2 instance	35
Troubleshooting	28
Collection of operational metrics	38
Operational metrics and alarms	39
Performance	41
Source code	42
Contributors	43

Revisions	44
Notices	45

Solution to specify elastic and dynamic cloud resources as objects that can be easily referenced within AWS Network Firewall rules

Publication date: *March 2022* ([last update](#): *June 2023*)

Traditionally, firewall products are not cloud aware, and addresses allocated to dynamic instances in the cloud cannot be predetermined. As a result, customers must create rules based on ranges covering entire AWS accounts, Amazon Virtual Private Clouds (Amazon VPCs), or subnets, at odds with the security principle of least-privilege (PoLP). Such rigid dependence on hard-coded entities detracts from the dynamic, flexible, and elastic nature of the cloud infrastructure. Moreover, the challenge of keeping firewall configurations up to date when referenced endpoints are modified or removed, necessitates a high-touch process between application teams and security operations staff.

The Dynamic Object and Rule Extensions for AWS Network Firewall solution provides an object abstraction around elastic and dynamic groups of AWS resources, so that these resources can be referenced within AWS Network Firewall (ANFW) rules, and continually and automatically synchronized as these resources scale in or out.

This implementation guide describes architectural considerations and configuration steps for deploying Dynamic Object and Rule Extensions for AWS Network Firewall in the Amazon Web Services (AWS) Cloud. It includes links to an [AWS CloudFormation](#) template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT architects, developers, DevOps, data analysts, and marketing technology professionals who have practical experience architecting in the AWS Cloud.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost to run this solution depends on the number of policies installed, the number of AWS Lambda functions and their running duration, and the number of Amazon EventBridge events published. As of the most recent revision, the estimated cost for running this solution with default settings in the US East (N. Virginia) Region without Open Policy Agent (OPA) activated is approximately **\$96.48**, and with OPA activated is approximately **\$137.48**. The option with OPA activated provides advanced validation for the API. Refer to [Architecture overview](#) for more details.


Note

To activate the rules managed by this solution an AWS Network Firewall (ANFW) instance is required. Refer to [AWS Network Firewall pricing](#) for more details about the pricing. The following cost estimation excludes ANFW instance pricing.

AWS service	Dimensions	Cost/month
AWS Lambda	8640 requests per month, Lambda memory: 3GB of memory	\$8.64
AWS CloudWatch Events	Number of custom/cross-account events (8640), number of Lambda functions (2), number of requests per function (8640 per month).	\$6.01
Amazon S3	0.1 GB bucket size	Negligible
NAT Gateway	0.5 GB data/hour, \$ 0.059/hour	\$43.13
DynamoDB	1GB storage, 200Kb average size for attributes	\$8.70
AWS Config	10000 resources	\$30.00
	Total (without OPA enabled):	\$96.48

AWS service	Dimensions	Cost/month
AWS Fargate*	0.5 vCPU, 2GB vMemory	\$22.00
Elastic Load Balancing (ELB)	1 * ALB, 2 new connections per second	\$19.00
	Total (with OPA activated; enableOpa = true):	\$137.48

*AWS Fargate is used with Amazon Elastic Container Service (Amazon ECS).

 **Note**

The cost associated with Amazon ECS on AWS Fargate and ELB only occurs with OPA activated, that is when configuration enableOpa is set to true.

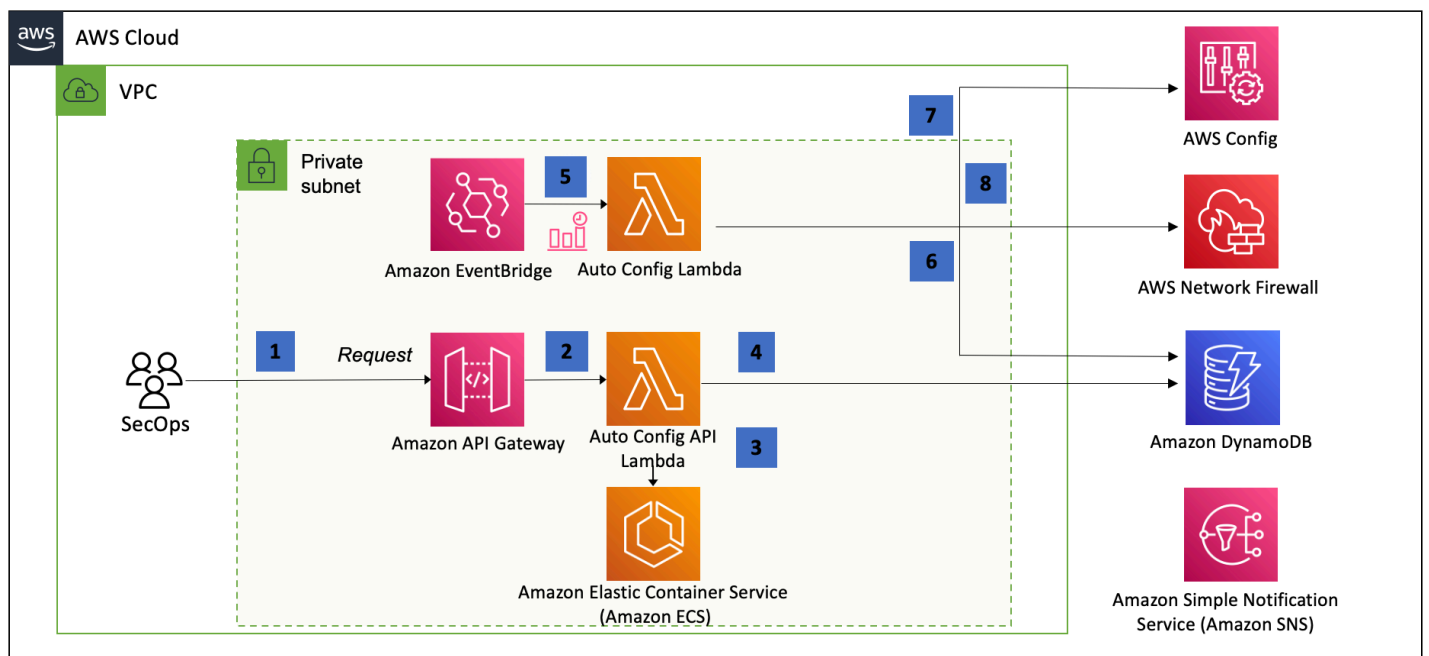
We recommend creating a [budget](#) through [AWS Cost Explorer](#) to help manage costs. Prices are subject to change. For full details, see the pricing webpage for each AWS service used in this solution.

Architecture overview

The Dynamic Object and Rule Extensions for AWS Network Firewall solution has a two-tier architecture, which consists of the business tier and data tier. The business tier is powered by a combination of [Amazon API Gateway](#) and [AWS Lambda](#). The business logic predominantly resides in AWS Lambda for managing the domain data and periodically running a Lambda function to keep data synchronized between the solution and ANFW instance. The data tier is underpinned by [Amazon DynamoDB](#) to store rule bundle, object, and rule data.

The architecture can be grouped into two logical components:

- Request orchestration
- Automatic resource and rule synchronization, and ANFW configuration



Dynamic Object and Rule Extensions for AWS Network Firewall solution architecture

1. The API Gateway provides the primary interface for the user to interact with this solution, including endpoints to manage the domain entities. Domain entities include rule, object, rule bundle, and list audit information. Refer to [API schema](#) in the GitHub repository for sample requests and information about updating the metadata.
2. The request is forwarded to a Lambda handler function.

3. (Optional) When `enableOpa = true`, a Lambda function invokes ECS-hosted OPA cluster to exercise validation on the request based on context. For example, Lambda function can validate if the requester is allowed to perform the `CreateObject` action.
4. Lambda issues request data to read from or write to domain entity tables in DynamoDB.
5. An [Amazon EventBridge](#) rule is scheduled to invoke the Auto Config Lambda function. The frequency is based on the `ruleResolutionInterval` configuration; the default value is 10 minutes.
6. The auto config Lambda function requests domain entity data such as rule bundle, rule, and object from Amazon DynamoDB.
7. The auto config Lambda function queries the [AWS Config aggregator](#) to resolve defined object referenced by rule in the solution.
8. The auto Config Lambda function sends an update request to ANFW.

Solution components

Amazon DynamoDB

The solution uses DynamoDB to save solution data as rule bundle, rule, object and audit. The data is used by the Lambda functions to update the underlying ANFW rules.

Amazon EventBridge

The solution uses the EventBridge rule to invoke Lambda functions periodically to keep the rules synchronized between the solution and the underlying ANFW rules.

AWS Config

The solution queries an AWS Config aggregator for the metadata of a given cloud resource, for example, the IP address for an EC2 instance.

AWS Network Firewall

The solution manages the rule group in Network Firewall and continuously updates the rules in rule groups according to the rules.

Amazon Simple Storage Service*

The solution creates one Amazon S3 bucket in your account which is used to host the OPA policy bundle.

Amazon Elastic Container Service*

The solution uses the Amazon Elastic Container Service (Amazon ECS) to host the OPA cluster to allow the Request Orchestrator Lambda to validate the requests' validity.

Note

*Amazon S3 and Amazon ECS are required only when `enableOpa` is set to `true`. Refer to [Update solution configuration](#) for more information.

AWS Lambda

The Dynamic Object and Rule Extensions for AWS Network Firewall solution uses Lambda functions to store data for rule bundle, rule, object, and audit information. Lambda functions also update ANFW periodically.

The solution deploys the following three Lambda functions:

Request orchestrator

A Lambda function that orchestrates the incoming user-initiated request from Amazon API Gateway. It performs request validation, and manages the domain data such as rule bundle, rule, object according to the request in Amazon DynamoDB.

Auto Config Scheduler

A Lambda function that is periodically triggered by Amazon EventBridge to find the rule bundle entity targeted at the same underlying resources. The interval is based on the configuration, refer to [Update solution configuration](#) for more information.

Auto Config

A Lambda function to resolve the rules and objects defined in rule bundles and translate them into standard ANFW rules.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This [shared model](#) reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit [AWS Cloud Security](#).

IAM roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users in the AWS Cloud. This solution creates the following IAM roles that grant AWS Lambda functions access to create regional resources.

LambdaExecutionRoles: Each Lambda function has a dedicated role to allow it to interact with Amazon DynamoDB, AWS Config, and AWS Network Firewall in a least privilege manner.

ObjectExtensionSecOpsAdminRole: Assumed by the customer to interact with this solution.

Network configuration

The solution is deployed in Amazon Virtual Private Cloud (Amazon VPC) with the Lambda functions in a private subnet. Traffic in and out of the private subnet is controlled by security groups. By default, the security group rules only allow inbound traffic from the private subnet to prevent unauthorized access to the data storage layer.

By default, the solution creates a new Amazon VPC with two subnets each in two availability zones. Each availability zone includes:

- One public subnet to install a NAT gateway, so that certain AWS services can be reached, for example, AWS Network Firewall.
- One private subnet to install ECS cluster (if `enableOpa` is set to `true`) and all Lambda functions, so that no data leaves the Amazon VPC (exclude the communication to and from AWS Network Firewall)
- Eight network interfaces are created to ensure the data is within the Amazon VPC.

Security groups

The security groups created in this solution are designed to control and isolate network traffic between the Lambda functions, Certificate Signing Request (CSR) instances, and remote virtual private network (VPN) endpoints. Once the deployment is up and running, we recommend that you review the security groups and further restrict access as needed.

Data protection

All data committed to the solution is encrypted at rest. This includes the data stored in Amazon S3 and DynamoDB.

Communication between the solution's different components is over HTTPS to ensure data is encrypted in transit, and within the VPC boundary.

By default, all S3 buckets for this solution come with the following configuration:

- All public access blocked
- Versioning activated
- Access log activated
- Encryption at rest by a key management service (KMS)-based customer managed key (CMK)

Additionally, S3 buckets are also configured with a default resource policy that deny all non-HTTPS requests to ensure data-in-transit encryption.

Domain concepts

Roles

Roles represents the operator's logical identity that is going to interact with this solution. Currently, this solution supports a single IAM role. This role represents a Security Operations (SecOps) user, or a user that would typically configure the ANFW instance.

Rule bundle

A rule bundle in the solution is an abstraction representing the underlying resource which aggregates the firewall rules. It represents an ANFW rule group associated with an individual role or team using the solution, and a collection of rules to be applied to the firewall.

Rule

A rule defines what kind of traffic is allowed or blocked, in respect to the objects it references. For instance, a rule could be defined to allow traffic from a static IP (an object) to a cloud resource (another object). A rule is an object abstraction which results in a Suricata rule being automatically applied to the ANFW rule group once the objects are resolved.

Object

An object is an abstraction of the resource which rules can reference. It can be a static resource, such as a fixed IP/CIDR, or dynamic cloud resource, such as an AWS ARN or tag. The solution currently supports the following types of objects:

Objects	Description
Amazon EC2 ARN	Resolves to the IP of the EC2 instance
Auto Scaling group ARN	Resolves to the IP of the EC2 instance in the ASG
Amazon Virtual Private Cloud ARN	Resolves to the CIDR of the VPC

Objects	Description
Amazon Virtual Private Cloud Subnet ARN	Resolves to the CIDR of the subnet
Tags	Resolves to the EC2 instances and AWS Lambda functions which contain matching tag values
Amazon EC2 security group ARN	Resolves to the IP of EC2 instances to which this security group is attached

Audit

An audit is a record for all mutation actions: create, update, delete (CRUD) upon rule bundle, rule, and object. It records the requestor, its action, requested content, and the result of the actions.

For more information, refer to [API schema](#) in the GitHub repository for supported object types.

Design considerations

Event-driven vs. continuous polling

This solution needs to resolve all objects for all the rules in the referenced group, which is not an incremental process. To resolve all objects for all the rules, introducing an event-driven mechanism would mean having to handle a race condition making the solution more complex, and the resolution less reliable. This would cause a high error rate on rule resolution. Therefore, the solution uses continuous polling as the main mechanism to resolve objects and rules.

Regional deployments

This solution uses the AWS Network Firewall service which is not currently available in all AWS Regions. You must launch this solution in an AWS Region where AWS Config and AWS Network Firewall is available. For the most current availability by Region, refer to the [AWS Regional Services List](#).

As of June 2022, this solution is supported in the following Regions:


Region ID	Region name
us-east-1	US East (N. Virginia)
us-west-2	US West (Oregon)
eu-west-2	Europe (London)
ap-south-1	Asia Pacific (Mumbai)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)

AWS CloudFormation template

To automate deployment, this solution uses the following AWS CloudFormation template which you can download before deployment:

[View template](#)

FirewallObjectExtensionSolutionStack.template: Use this template to launch the solution and all associated components. The default configuration deploys Amazon S3, AWS Lambda, and resources needed for the solution but you can customize the template to meet your specific needs.

 **Note**

AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

Deployment

Before you deploy this solution, review the architecture and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 15 mins

Prerequisites

Install tools

Install the following tools.

- The latest version of the [AWS CLI](#) (2.2.37 or newer), installed and configured.
- The latest version of the [AWS CDK](#) (1.116.0 or newer).
- Make 3.5.0 or newer. For Mac OS X, run `brew install make`; for Amazon Linux 2, run `yum install make`.
- (Optional) [Docker](#) 19.0.0 or newer (required when `enableOpa` is set to `true`).
- A CDK bootstrapped AWS account, refer to [Bootstrapping](#) for more information.
- Node.js version 14, refer to [Getting started](#) for more information.

Activate AWS Config and AWS Network Firewall

- Ensure AWS Config aggregator is configured and accessible to the solution. The Config aggregator can operate in the same account to which the solution will be deployed to, or can be accessed using the `crossAccountConfigReadOnlyRole` role when located in another account.
- Create at least one Network Firewall rule group before installing this solution.

Install AWS Certificate Manager (ACM) certificate

- (Optional) The ACM certificate is required only when `enable0pa` is set to `true`. A certificate is required to import into ACM to allow the solution to activate transport layer security (TLS) in Application Load Balancer.

Deployment overview

Deploying this architecture on AWS includes the following steps. For detailed instructions, follow the links for each step.

[Step 1. Update solution configuration](#)

[Step 2. Build and deploy the stack](#)

1. (Optional) Cross-account support.
2. (Optional) Deploy to existing VPC.

Step 1. Update solution configuration

The solution can be configured to suit different customer requirements for rule synchronization delay and specific cloud resources. All configuration items can be configured in the `cdk.context.json` file located in the `source` folder.

The supported configuration and their default values are as follows:

Note

None of the following properties are mandatory during configuration. If no customer value is provided, the default value of the configuration is used.

Configuration	Description	Default value	Value type
<code>networkFirewallRuleGroupNamePattern</code>	Allowed Network Firewall rule group pattern	<code>default-anfwconfig-rule-*</code>	string
<code>defaultAggregatorName</code>	AWS Config aggregator name	<code>org-replicator</code>	string

Configuration	Description	Default value	Value type
	used by this solution, when creating a new rule group. If no aggregator is provided, defaultAggregator will be assigned to the rule group		
ruleResolutionInterval	The interval rules are resolved and applied into Network Firewall	10 mins	integer, min value 5, max value 60
failureNotificationTargetEmails	The email address for sending notifications. Once rule resolution failure happens, customer can add their email to the SNS topic manually later on	[]	list of strings
apiGatewayType	The type of API gateway	private	edge private
enableOpa*	(OPA specific configuration) Enable OPA cluster to validate rule and object mutation requests	FALSE	true false
certificateArn*	(OPA specific configuration) ACM certification for the ALB	used when enableOpa is set to true	

Configuration	Description	Default value	Value type
crossAccountConfig ReadOnlyRole**	AWS IAM read-only role in the account in which the AWS config is activated	NULL, target at solution installation account	string
crossAccount NetworkFirewall ReadWriteRole**	AWS IAM read/write role account in which the AWS network firewall instance is setup	NULL, target at solution installation account	string

Override default configuration values

There are two ways to update the default values provided by the solution:

1. Update the `cdk.context.json` file.
 - a. Go to the solution's `source` folder.
 - b. Locate the `cdk.context.json` file.
 - c. Update the value of any supported configuration listed in the above table.
 - d. Follow the steps in [Build and deploy the stack](#) section.
2. Passing the parameter from command line.
 - a. Follow step 7.b in [Build and deploy the stack](#) section.

Step 2. Build and deploy the stack

1. Clone the solution source code from its [GitHub repository](#).
2. Open the terminal and navigate to the folder created in step 1.
3. Navigate to the `source` folder.
4. Run the following command to install a global npm plugin to allow cross-account deployment:

```
npm install -g cdk-assume-role-credential-plugin
```

5. Run `npm run install:all`.
6. Run `npm run all`.
7. Run `cdk synth`.
 - a. (Optional for customize configuration value), run `cdk synth -c <Configuration> <Value>`

For example, to override `ruleResolutionInterval`, run:

```
cdk synth -c ruleResolutionInterval=5
```

This would set the `ruleResolutionInterval` from default 10 minutes to five minutes.

8. Run `cdk deploy`.
 - a. Ensure docker daemon is running (if OPA is activated).

`cdk boot strap` is completed for the deployment account.

(Optional) Activate cross-account access

This section is required only if the AWS Config and/or AWS Network Firewall instance is in a different account in which the solution is installed.

Activate cross-account access for AWS Config

To allow cross-account access, create a trusted role in the hosting account in which AWS Config is going to be used.

1. Update the solution's configuration `crossAccountConfigReadOnlyRole` to `arn:aws:iam::<TARGET_ACCOUNT>:role/ConfigReadOnlyRole` where `TARGET_ACCOUNT` is the account AWS Config is running.
2. Redeploy the solution. For details, refer to [Build and deploy the stack](#).
3. Deploy cross account role.
 - a. Sign in to the [AWS Management Console](#) of the `TARGET_ACCOUNT`.
 - b. Select the button to launch the `deployment-prerequisites/cross-account-role-config-stack.yaml` AWS CloudFormation template.

- c. Choose **Next**.
- d. Update the stack name to `Object-extension-AWS-config-role-stack`.
- e. Update `solutionInstalledAccount` to the target account.
- f. Choose **Next**.
- g. Choose **Create stack** to deploy the stack.

The stack deploys the following resources to the target account.

1. Cross-account trust IAM role `ConfigReadOnlyRole` in the target account.
2. Trust relationship to the solution installation account.

Activate cross-account access for AWS Network Firewall

This solution also supports cross-account access for AWS Network Firewall.

1. Update the solution's configuration `crossAccount NetworkFirewall ReadWriteRole` to `arn:aws:iam::<TARGET_ACCOUNT>:role/NetworkFirewallReadWriteRole` where `TARGET_ACCOUNT` is the account AWS Config is running.
2. Redeploy the solution. For details, refer to [Build and deploy the stack](#).
3. Deploy cross-account role.
 - a. Sign in to the [AWS Management Console](#) of the `TARGET_ACCOUNT`.
 - b. Select the button to launch the `deployment-prerequisites/cross-account-role-anfw-stack.yaml` AWS CloudFormation template.
 - c. Choose **Next**.
 - d. Update stack name to `Object-extension-AWS-network-firewall-role-stack`.
 - e. Update `solutionInstalledAccount` to the target account.
 - f. Choose **Next**.
 - g. Choose **Create stack** to deploy the stack.

(Optional) Deploy to an existing VPC

If your VPC is created outside your CDK app, you can use the configuration `importVpcDetails` to import your existing VPC to be used for this solution. The solution will leverage the CDK's `Vpc.fromVpcAttributes` way to import VPCs. There are three requisites to import existing VPCs.

- Every subnet group in the VPC must have a subnet in each availability zone (AZ) (for example, each AZ must have both a public and private subnet). Asymmetric VPCs are not supported.
- Public subnet requires a NAT gateway so that services in private subnet can use a certain AWS service. For example, AWS Network Firewall.
- All configuration parameters `vpcId`, `availabilityZones`, `publicSubnetIds`, `privateSubnetIds`, `vpcCidrBlock` must either be known at synthesis time, or they must come from deploy-time list parameters whose deploy-time lengths are known at synthesis time. The parameters must also match the target VPC exactly. For the usage of public and private subnets, refer to [Network configuration](#).
- Imported VPC must activate the following VPC endpoints:
 - EC2
 - EC2 Message
 - Lambda
 - SNS
 - KMS
 - CloudWatch
 - AWS Config
 - If `enableOpa` is set to `true`, Amazon ECR, ECR Docker, Elastic Load Balancer endpoints are required.

Example: Using an existing VPC

Update the configuration in `cdk.context.json`.

```
{
  "networkFirewallRuleGroupNamePattern": "anfwconfig-*",
  "defaultAggregatorName": "org-replicator",
  "apiGatewayType": "edge",
  "enableOpa": false,
  "importVpcDetails": {
    "vpcId": "vpc-0d2c9296c3b26e396",
    "availabilityZones": ["ap-southeast-2a", "ap-southeast-2b"],
    "publicSubnetIds": ["subnet-019358ac809ad81a2", "subnet-0fea75ee8255c55d5"],
    "privateSubnetIds": ["subnet-09ec1f4a0336f6431", "subnet-009dca942826cf5b1"],
    "vpcCidrBlock": "172.31.0.0/16"
  }
}
```

```
}
```

To verify all the roles and roles' access are set up correctly, refer to [Interaction steps](#) to setup a rule bundle and rule, and to test the resolution.

Interaction steps

This section details how to configure AWS Network Firewall to interoperate with this solution, create example objects, and a rule that references these objects.

- Create an ANFW rule group to hold the resultant rules created by the solution.
- Create a rule bundle within the solution, to hold the dynamic rules defined by the security operator.
- Create objects to be referenced within the solution rules.
- Create rules within the rule bundle, which contain references to the objects created earlier.
- The solution populates the ANFW rule group with the defined rules as objects are resolved through querying the AWS Config aggregator.

Identify a host machine with which to interact with the solution

If you are interacting with the solution from a machine within AWS, rather than a local desktop, complete the following steps.

Note

The following section is required if configuration `apiGatewayType` is set to `private`. You must complete the following steps while logged into EC2 via Session Manager.

1. Sign in to the [AWS Systems Manager Session Manager](#) console.
2. Choose **Start Session**.
3. Select the target instance. For details, refer to [Create an EC2 instance](#).
4. Choose **Start Session**.

If configuration `apiGatewayType` is set to `edge`, complete the following step on your local machine to interact with the solution's Edge-optimized API endpoint.

If configuration `apiGatewayType` is set to `private` (default), complete the following steps from the EC2 Session Manager's connection opened in the previous step to interact with the solution's private API endpoint.

Install the prerequisite tools

1. Install JQ. Refer to [Install JQ](#) for more information.
2. Install awscurl. Refer to [Install awscurl](#) for more information.
3. On Mac OS X, install GNU getopt. Refer to [Install GNU getopt](#) for more information.

Create a Network Firewall rule group

1. Sign in to the AWS Management Console.
2. Select **VPC**, then select **Network Firewall Rule Groups** in the left navigation menu.
3. On the Network Firewall rule groups page, choose **Create Network Firewall rule group**.
4. Select **Stateful rule group**.
5. Make sure the Stateful rule group name is aligned with the pattern in configuration section's `networkFirewallRuleGroupNamePattern`. If the pattern is `default-ANFW-CONFIG-*`, then the rule group name should be `default-ANFW-CONFIG-rule1`, so that this solution can request the resource from ANFW correctly.
6. Fill **Capacity** value, we recommend at least 15,000 for this solution.
7. Select **IP**, then add a default rule.
8. Choose **Create stateful rule group**.

Record the name of the stateful rule group you created. You will use it in [Use CLI tool to create solution rule bundle entity](#).

The following section describes the process for creating a new object and a rule group, then adding a rule in the new rule group to reference the object.

Use CLI tool to create solution rule bundle entity

1. Ensure you are using GNU getopt command line tool. This is the default on Linux. If you are using the CLI on Mac OS X, refer to [Install GNU Getopt](#) for instructions about installing GNU getopt. Ensure you have set the environment variable `GNU_GETOPT_PATH` if GNU getopt is not in your path already.

```
export GNU_GETOPT_PATH=/usr/local/opt/gnu-getopt/bin/getopt
```

2. Run the following command to set account this solution is installed:

```
export ACCOUNT_NUMBER=1234567
```

3. Run the following command to set the API_ENDPOINT that the solution created during installation:

```
export API_ENDPOINT=https://yourAPIendpoint.execute-api.ap-southeast-2.amazonaws.com/prod
```

The API_ENDPOINT can be found in the CloudFormation console as below

4. Run the following command to set ROLEARN to allow the user to interact with API gateway of this solution:

```
export ROLEARN=$(aws iam get-role --role-name  
"ObjectExtensionSecOpsAdminRole"-<region> | jq -r .Role.Arn)
```

5. Assume to SecOps role using ROLEARN from above.

```
export ASSUMEROLE=$(aws sts assume-role --role-arn $ROLEARN --role-session-name  
DeviceClient --duration-second 3600)  
export AWS_ACCESS_KEY_ID=$(echo $ASSUMEROLE | jq -r .Credentials.AccessKeyId)  
export AWS_SECRET_ACCESS_KEY=$(echo $ASSUMEROLE | jq -  
r .Credentials.SecretAccessKey)  
export AWS_SESSION_TOKEN=$(echo $ASSUMEROLE | jq -r .Credentials.SessionToken)
```

6. Run the following command to set AWS_REGION to ensure your credentials are used in the same region that the API is hosted:

```
export AWS_REGION=<solution aws region>
```

7. Run the following command to set RULE_GROUP_NAME to the rule group you created earlier:

```
export RULE_GROUP_NAME=<name of stateful rule group from previous step>
```

Example:

```
export RULE_GROUP_NAME=anfwwconfig-testrulegroup-demo-001
```

Run the following commands:

```
cd scripts
```

```
./aoe.sh create-rule-bundle --rule-bundle-id example-rule-bundle-1 --rule-bundle-  
description "Example Solution Rule Bundle" --rule-group-arn arn:aws:network-  
firewall:ap-southeast-2:${ACCOUNT_NUMBER}:stateful-rulegroup/${RULE_GROUP_NAME} --  
rule-bundle-owner-group "${ROLEARN}"
```

Expected output:

```
{  
  "ruleBundleId": "example-rule-bundle-1"  
}
```

Create an object to represent the underlying cloud resource

1. Create an EC2 instance, find its instance id, and set the environment EC2_ARN.

```
export EC2_ARN="arn:aws:ec2:ap-southeast-2:${ACCOUNT_NUMBER}:instance/<instance  
id>"
```

2. Create an object reference to this Arn with the following command:

```
./aoe.sh create-object --object-id "MyExampleObject" --object-type "Arn" --object-  
value ${EC2_ARN}
```

Expected output:

```
{  
  "object": {  
    "id": "MyExampleObject",  
    "type": "Arn",  
    "value": "arn:aws:ec2:ap-southeast-2:111111111:instance/i-0564e8a206c6fb237",  
    "createdBy": "arn:aws:sts::111111111:assumed-role/  
ObjectExtensionSecOpsAdminRole/DeviceClient",  
    "lastUpdated": "2021-10-15T00:32:38.723Z"  
  }  
}
```

3. Create one additional fixed object with the following command. This is an object to represent IP 172.16.1.20 and port 1234:

```
./aoe.sh create-object --object-id "OnPremServer" --object-type "Address" --object-value 172.16.1.20
```

Expected output:

```
{
  "object": {
    "id": "OnPremServer",
    "type": "Address",
    "value": "172.16.1.20",
    "createdBy": "arn:aws:sts::<ACCOUNT_ID>:assumed-role/
ObjectExtensionSecOpsAdminRole/DeviceClient",
    "lastUpdated": "2021-10-07T01:43:29.854Z"
  }
}
```

Create a rule in the rule bundle

1. Use the following command to create a new rule, which references both objects created in the previous step.

```
./aoe.sh create-rule --rule-bundle-id example-rule-bundle-1 --rule-protocol tcp --rule-action pass --rule-source OnPremServer --source-port-type SinglePort --source-port-value 1234 --rule-destination MyExampleObject --destination-port-type Any
```

Expected output:

```
{
  "rule": {
    "protocol": "tcp",
    "action": "pass",
    "source": "OnPremServer",
    "sourcePort": {
      "type": "SinglePort",
      "value": "1234"
    },
    "destination": "MyExampleObject",
    "destinationPort": {
```

```
    "type": "Any"
  },
  "status": "PENDING",
  "ruleBundleId": "example-rule-bundle-1",
  "lastUpdated": "2021-10-15T00:37:07.576Z",
  "id": "94bbbb56-fd3d-4f80-a74f-749b4d907fb9",
  "version": 0
}
}
```

2. At this point, the solution will process the rule in the next cycle and automatically configure ANFW with a rule that reflects the resolved addresses of both objects. Refer to the ANFW console.

Troubleshooting

If you need help with this solution, contact Support to open a support case for this solution.

Contact Support

If you have [AWS Developer Support](#), [AWS Business Support](#), or [AWS Enterprise Support](#), you can use the Support Center to get expert assistance with this solution. The following sections provide instructions.

Create case

1. Sign in to [Support Center](#).
2. Choose **Create case**.

How can we help?

1. Choose **Technical**.
2. For **Service**, select **Solutions**.
3. For **Category**, select **Other Solutions**.
4. For **Severity**, select the option that best matches your use case.
5. When you enter the **Service**, **Category**, and **Severity**, the interface populates links to common troubleshooting questions. If you can't resolve your question with these links, choose **Next step: Additional information**.

Additional information

1. For **Subject**, enter text summarizing your question or issue.
2. For **Description**, describe the issue in detail.
3. Choose **Attach files**.
4. Attach the information that AWS Support needs to process the request.

Help us resolve your case faster

1. Enter the requested information.
2. Choose **Next step: Solve now or contact us**.

Solve now or contact us

1. Review the **Solve now** solutions.
2. If you can't resolve your issue with these solutions, choose **Contact us**, enter the requested information, and choose **Submit**.

Uninstall the solution

You can uninstall the Dynamic Object and Rule Extensions for AWS Network solution from the AWS Management Console or by using the AWS Command Line Interface. You must manually delete S3, Lambda, and other resources created by this solution. AWS Solutions Implementations do not automatically delete DynamoDB tables in case you have stored data to retain.

Using the AWS Management Console

1. Sign in to the [AWS CloudFormation console](#);
2. On the **Stacks** page, select this solution's installation stack.
3. Choose **Delete**.

Using the AWS Command Line Interface (CLI)

To uninstall the solution

- Run `cdk destroy` from the `sources` folder, or
- Delete the stack from the CloudFormation console. Note that the cross-account access stacks also need to be deleted for this method.

Note

For data retention and audit purpose the following resources will not be removed.

- OPA policy bucket and its encryption key in KMS.
- All four domain data DynamoDB tables (including the table for rule bundle, rule, object and audit) and their encryption keys in KMS.

Additional resources

AWS services

- [AWS CloudFormation](#)
- [Amazon S3](#)
- [AWS Lambda](#)
- [Amazon DynamoDB](#)
- [Amazon API Gateway](#)
- [Amazon Elastic Container Service](#)

Operation and customization

Data backup and restore

All DynamoDB tables have point-in-time recovery (PITR) activated by default. Perform these steps in a testing environment before applying to a production environment.

1. Sign in to the [Amazon EventBridge console](#), then select **Amazon EventBridge**.
2. Select **Rules** and find the `FirewallObjectExtensionSo-autoconfigTriggerRule*` rule.
3. Select the rule and choose **Disable** to temporarily disable the rule. Refer to [Deleting or Disabling a CloudWatch Events Rule](#) for more information.
4. Navigate to the [DynamoDB console](#). Select **Backups** in the left navigation menu and find the target backup. For example, `rulesbackup`.
5. Export backup table data into S3, refer to [Exporting and Importing DynamoDB Data Using AWS Data Pipeline](#) for more information.
6. For restore, import S3 data to the `RuleExtensionsRuleTable` solution table. Refer to [Exporting and Importing DynamoDB Data Using AWS Data Pipeline](#) for more information.
7. Repeat steps 1 through to 6 until all tables are restored.
8. Before restore, navigate to the [Amazon EventBridge console](#). Select **Rules**, then select the `FirewallObjectExtensionSo-autoconfigTriggerRule*` rule, and choose **Enable**.

Allow the rules to be propagated to ANFW, then verify the ANFW rules number is correct. Also, verify the rule status by calling the solution's API `rulebundles/<id>/rules`.

AWS resources and service limitations

The Dynamic Object and Rule Extensions for AWS Network Firewall solution leverages the following AWS services and is subject to their respective limitations.

- AWS Identity & Access Management
- Amazon Virtual Private Cloud
- Amazon API Gateway
- AWS Lambda

- Amazon DynamoDB
- Amazon CloudWatch Synthetics
- Amazon CloudWatch
- AWS Network Firewall
- Amazon Elastic Container Service
- AWS Config
- AWS X-Ray

API schema

Visit our [GitHub repository](#) for the full schema and API documentation.

Install JQ

JQ is used to parse command results from the CLI tool.

To install on Amazon Linux 2:

```
sudo yum install jq
```

To install on Mac OS X:

- Install [Homebrew](#).
- Once Homebrew is installed, run the following command:

```
brew install jq
```

Install AWS curl

AWS curl is used to call Sigv4 protected AWS API endpoints using AWS credentials.

To install on Amazon Linux 2:

```
pip3 install --user awscurl
```

To install on Mac OS X:

- a. Install [Homebrew](#).
- b. Once Homebrew is installed, run the following command:

```
brew install awscli
```

Install GNU Getopt

Note

GNU Getopt is not required on Amazon Linux 2.

To install on Mac OS X:

1. Install [Homebrew](#).
2. Once Homebrew is installed, run the following command:

```
brew install gnu-getopt
```

3. Set GNU_GETOPT_PATH for use by the CLI tool.

```
export GNU_GETOPT_PATH=/usr/local/opt/gnu-getopt/bin/getopt
```

Customize OPA policy

1. Add a new OPA policy under opa/packages/<folder>/, <folder> could be objects or rules. In this example, the policy ID is new_customer_policy, and the new policy file is opa/packages/objects/object_new_customer_policy.rego.
2. Register the new policy by using the following code:

```
FF_CUSTOMER_REGISTERED_POLICIES := [{  
  "level": "mandatory",  
  "packageId": "objects",  
  "policyId": "new_customer_policy_a",  
  "parameters": {},  
}]
```

```
  ]]  
  availablePolicySet[p] {  
    p := FF_CUSOMTER_REGISTERED_POLICIES  
  }  
}
```

3. (Optional) We recommend adding OPA policy unit test for the newly added policy. Refer to our [GitHub repository](#) for an example of the test.
4. Refer to the [README.md](#) file to build the bundle.
5. Redeploy this solution.

Create an EC2 instance

Create an Amazon EC2 instance to access the API Gateway endpoint in Amazon VPC.

1. Sign in to the [EC2 console](#).
2. On the Instances page, choose **Launch Instances**.
3. Select the **Amazon Linux 2 AMI (HVM), SSD Volume Type** instance.
4. On the Choose an Instance Type page, select `t2.micro`, and then choose **Review and Launch**.
5. Select **Edit instance details**, and choose the following configuration:

```
Network: The VPC ID created by the solution.  
Subnet: Firewall1ObjectExtensionSolutionStack/object-extension-Vpc/  
PrivateSubnetASubnet2
```

6. Choose **Review and Launch**.
7. Review your Instance launch and choose **Launch**.
8. In the Select an existing key pair or create a new key pair box, choose **Proceed without a key pair** and check the **I will not be able to connect to this instance unless you use EC2 Instance Connect or know the password built into the AMI** box.
9. Choose **Launch Instances**.

Troubleshooting

If you need to troubleshoot issues with the deployed solution in case unexpected network and/or infrastructure change occurs. The following are basic diagnosis and recovery steps for some common issues.

API gateway return 504 or 503 for all requests

1. Sign in to the [CloudWatch console](#), then select **Dashboards**.
2. Select **RuleExtensionServiceDashboard**.
3. Check the alarms. The alarm reason is listed in **Details**.
4. Check **Synthetics canaries**. The logs section contains the error reasons in case of a failure.
5. Navigate to the AWS Console, then select **API Gateway**.
6. Select **Lambda**, then select **Monitor**.
7. Check log for Lambda, search for log level ERROR.
8. Get the `trace_id` from the ERROR log. For example, `trace_id = x_1123456`.
9. Navigate to the [CloudWatch console](#), then select **Trace** and enter the `trace_id`.

The next page will point to the error where it occurs.

Alternatively, if you cannot find `trace_id` from examining the logs in step 5, you can query `trace_id` by navigating to the [CloudWatch console](#), then select **Trace** and in the query section, enter:

```
service(id(name: "YOU_DEPLOYED_SERVICE_NAME" , type:
"AWS::Lambda::Function")) {fault = true}
```

Rule resolution failure

If Rule resolution fails with the rule in failed status and displays a failure reason.

1. Sign in to the [CloudWatch console](#), choose **Dashboards**, then select **RuleExtensionServiceDashboard**.
2. Check the alarms. The reason of the alarm is listed under **Details**.
3. Navigate to the [Lambda console](#) and search `Firewall1ObjectExtensionSo-autoconfigapiautoConfig`.
4. Navigate to the [API Gateway console](#) and choose **Lambda** then select **Monitor**.
5. Check log for Lambda, search for log level ERROR.
6. Get the `trace_id` from the ERROR log. For example, `trace_id = x_1123456`.
7. Navigate to the [CloudWatch console](#) then select **Trace**, and enter the `trace_id`.

Error: unauthorized access

This is a common error, for example, "User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: networkfirewall:DescribeFirewall on resource: exampleFirewall"

This is usually caused by an incorrect configuration of AWS Network Firewall rule group name. This solution can only access the pattern of AWS Network Firewall rule group name specified in the `networkFirewallRuleGroupNamePattern` configuration. Therefore, ensure the pattern is correct in respect to the actual name in AWS Network Firewall web console. Refer to [Update solution configuration](#) for more information.

Error: capacity exceeded

This is usually caused by an insufficient rule group capacity in respect to the number of rules defined in the system. Refer to [Performance](#) for more information.

1. Sign in to the [AWS Lambda console](#) and search for `FirewallObjectExtensionSo-autoconfigautoConfigSche`. Stop the trigger Event bridge, so that the rule resolution stops.
2. Navigate to the AWS Network Firewall console and select **Network Firewall rule groups**, then select your rule group.
3. Delete the group in full capacity. Run the following command to use the `aoe.sh` script that comes with this solution.

```
./aoe.sh get-rule --rule-bundle-id <id> --rule-id <id>
```

4. Create a new rule group with the same name but increase the capacity by 50% (recommended), or as appropriate.
5. Restart event bridge that was stopped in step 1.

For detailed troubleshooting steps for each AWS service used in this solution, refer to [Troubleshooting resources](#).

Collection of operational metrics

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected and sent to AWS:

- **Solution ID:** AWSSOLUTION/SO0196/1.0.0
- **Unique ID (UUID):** Randomly generated, unique identifier for each Dynamic Object and Rule Extensions for AWS Network solution deployment
- **Timestamp:** Data-collection timestamp

AWS owns the data gathered through this survey. Data collection is subject to the [AWS Privacy Policy](#). To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

1. Download the [AWS CloudFormation template](#) to your local hard drive.
2. Open the AWS CloudFormation template with a text editor.
3. Modify the AWS CloudFormation template mapping section from:

```
AnonymousData:  
  SendAnonymousData:  
    Data: Yes
```

to:

```
AnonymousData:  
  SendAnonymousData:  
    Data: No
```

4. Sign in to the [AWS CloudFormation console](#).
5. Select **Create stack**.
6. On the **Create stack** page **Specify template** section, select **Upload a template file**.
7. Under **Upload a template file**, choose **Choose file** and select the edited template from your local drive.
8. Choose **Next** and follow the steps in the [Deployment](#) section of this guide.

Operational metrics and alarms

This solution has a dash to list all the metrics for monitoring the core Lambdas and DynamoDB tables.

1. Navigate to the [CloudWatch console](#).
2. Select dashboard with prefix RuleExtensionServiceDashboard.

The metrics includes lambda invoke latency, API Gateway call response status and Error, DynamoDB capacity and latency.

This solution tolerates underlying system failure to prevent catastrophic cascade effect to your configured rules and firewall configurations. The primary design principle is to preserve the rule status in the database and AWS Network Firewall if underlying system failure happens. Once the impaired systems back up, the rules can sync up to the network firewall.

The following table lists potential service failures and the handling mechanism in place:

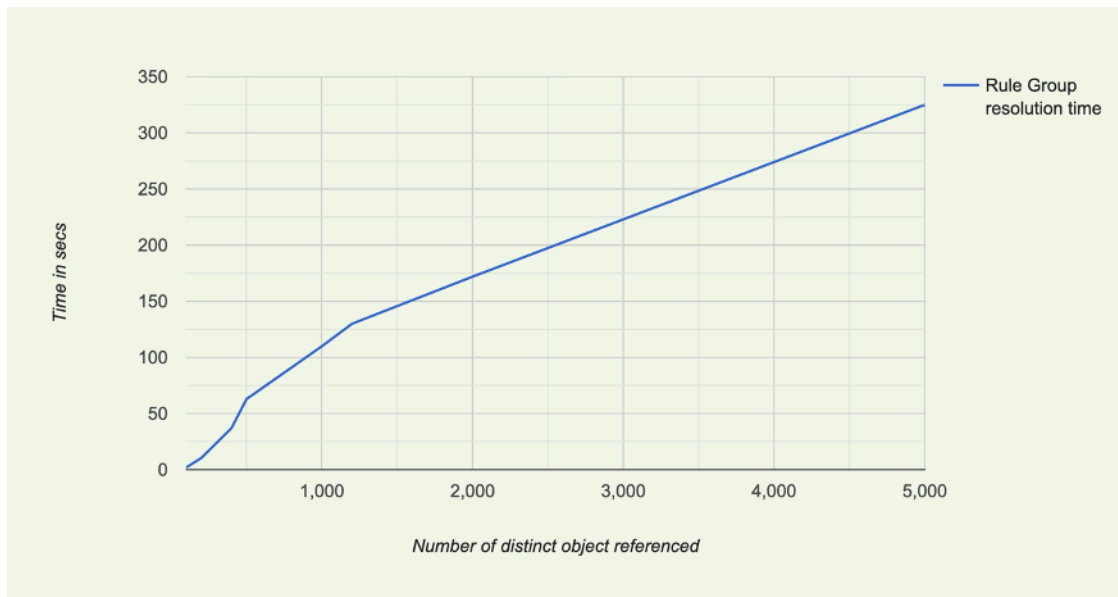
Service Name	Failure Type	Handling Mechanism	Behaviours once service back online
AWS Config	Unreachable	Send notification to registered email, trigger alarm in CloudWatch	All the rules will be re-evaluated and applied to AWS Network Firewall, no rules will be marked as failed (because of the AWS Config failure) during the process
	Internal Error	Send notification to registered email, trigger alarm in CloudWatch	*

Service Name	Failure Type	Handling Mechanism	Behaviours once service back online
AWS Network Firewall	Unreachable	Error rate alarm in CloudWatch	*
	Internal Error	Error rate alarm in CloudWatch	*
AWS EventBridge	Fail to send trigger event	Invocation alarm triggers in CloudWatch	*
AWS DynamoDB	Unreachable	Trigger alarm in CloudWatch	*

Find the alarms configured for this solution by navigating to the CloudWatch console and selecting Alarms. By default, the alarms' action is to send an email to the registered email address. Refer to [Update solution configuration](#) for information about how to configure alarms.

Performance

The Dynamic Object and Rule Extensions for AWS Network solution offers *near* real-time resolution of AWS resources encapsulated by custom defined objects. However, it is important to note, that the resolution time is a function of the total number of distinct objects being referenced across all rules within the rule group. As the number of referenced objects increases, the resolution time also increases linearly as shown in the graph below.



Rule group resolution time with respect to number of distinct objects referenced

To ensure performance of the solution, the capacity of the rule group ARN must be configured accordingly. We recommend setting the rule group capacity to at least 15,000.

Source code

Visit our [GitHub repository](#) to download the source files for this solution and to share your customizations with others. The Dynamic Object and Rule Extensions for AWS Network solution templates are generated using the [AWS Cloud Development Kit \(AWS CDK\)](#). Refer to the [README.md](#) file for additional information.

Contributors

- Yang Yang
- Nigel Brittain
- Deenadayaalan Thirugnanasambandam
- Tim O'Hare
- Hafiz Saadullah
- Swapnil Ogale

Revisions

Date	Change
March 2022	Initial release
June 2022	Release version 1.1.0: Minor updates and bug fixes. For more information about version 1.1.0, refer to the CHANGELOG.md file in the GitHub repository.
June 2023	<ul style="list-style-type: none">• Mitigated impact caused by new default settings for S3 Object Ownership (ACLs disabled) for all new S3 buckets. For more information, refer to the CHANGELOG.md file in the GitHub repository.• Added support for Node.js version 16.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Dynamic Object and Rule Extensions for AWS Network solution is licensed under the terms of the [Apache License, Version 2.0](#).