Cloud API on AWS v3.2.1

Created on: Apr 30, 2025

Notice

Copyright

Table of Contents

Chapter 1

Introduction to this Guide

This document describes the high-level architecture of Protegrity Cloud API on AWS, installation procedures, and performance guidance. This document focuses on Protegrity specific aspects and should be consumed in conjunction with corresponding AWS documentation.

This guide may also be used with the *Protegrity Enterprise Security Administrator Guide*, which explains the mechanism for managing the data security policy.

Chapter 2

Overview

2.1 Solution Overview

Cloud API on AWS is a cloud-native, serverless product for fine-grained data protection. This enables the invocation of Protegrity data protection cryptographic methods in cloud-native serverless technology. The benefits of serverless include rapid auto-scaling, performance, low administrative overhead, and reduced infrastructure costs compared to a server-based solution.

This product provides a data protection API end-point for clients. The product is designed to scale elastically and yield reliable query performance under extremely high concurrent loads. During idle use, the serverless product will scale completely down, providing significant savings in Cloud compute fees.

Protegrity utilizes a data security policy maintained by an Enterprise Security Administrator (ESA), similar to other Protegrity products. Using a simple REST API interface, authorized users can perform both de-identification (protect) and re-identification (unprotect) operations on data. A user's individual capabilities are subject to privileges and policies defined by the Enterprise Security Administrator.

2.2 Analytics on Protected Data

Protegrity’s format and length preserving tokenization scheme make it possible to perform analytics directly on protected data. Tokens are join-preserving so protected data can be joined across datasets. Often statistical analytics and machine learning training can be performed without the need to re-identify protected data. However, a user or service account with authorized security policy privileges may re-identify subsets of data using the Cloud API service.

2.3 Features

Cloud API on AWS incorporates Protegrity’s patent-pending vaultless tokenization capabilities into cloud-native serverless technology. Combined with an ESA security policy, the protector provides the following features:

- Role-based access control (RBAC) to protect and unprotect (re-identify) data depending on the user privileges.
- Policy enforcement features of other Protegrity application protectors.

For more information about the available protection options, such as, data types, Tokenization or Encryption types, or length preserving and non-preserving tokens, refer to *Protection Methods Reference Guide .*

2.4 Deployment Architecture

The product will be deployed in the Customer's AWS account. The product incorporates Protegrity's vaultless tokenization engine within an AWS Lambda Function. The encrypted data security policy from an ESA is deployed as a static resource through an Amazon Lambda Layer. The policy is decrypted in memory at runtime within the Lambda. This architecture allows Protegrity Serverless to be highly available and scale very quickly without direct dependency on any other Protegrity services.

The product exposes a remote data protection service. Each REST request includes a micro-batch of data to process and the data element type. The function applies the data security policy including user authorization and returns a corresponding response.

When used with an Enterprise Security Administrator (ESA) application, the security policy is synchronized through another serverless component called the Protegrity Policy Agent. The agent operates on a configurable schedule, fetches the policy from the ESA, performs additional envelope encryption using Amazon KMS, and deploys the policy into the Lambda Layer used by the serverless product. This solution can be configured to automatically provision the static policy or the final step can be performed on-demand by an administrator. The policy takes effect immediately for all new requests. There is no downtime for users during this process.

The following diagram shows the high-level architecture described above.

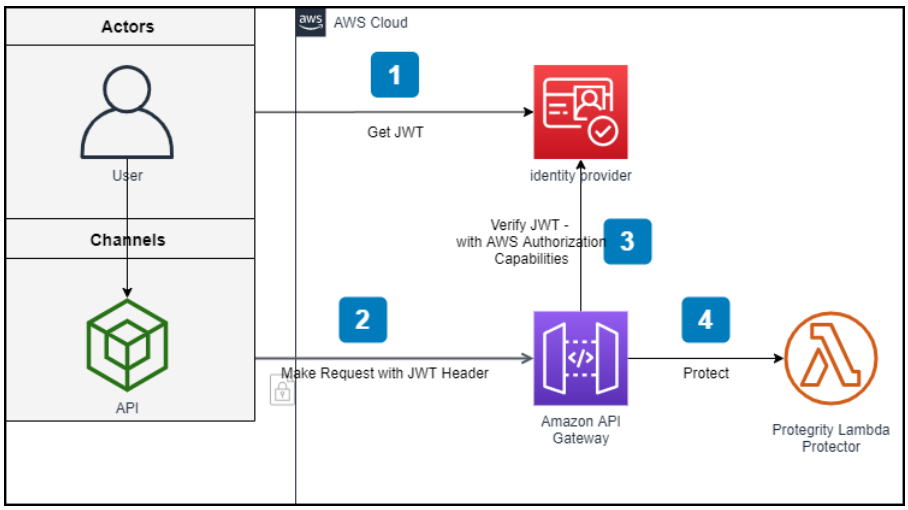


Figure 2-1: Cloud Protect API high-level architecture

The following diagram shows a reference architecture for synchronizing the security policy from ESA.

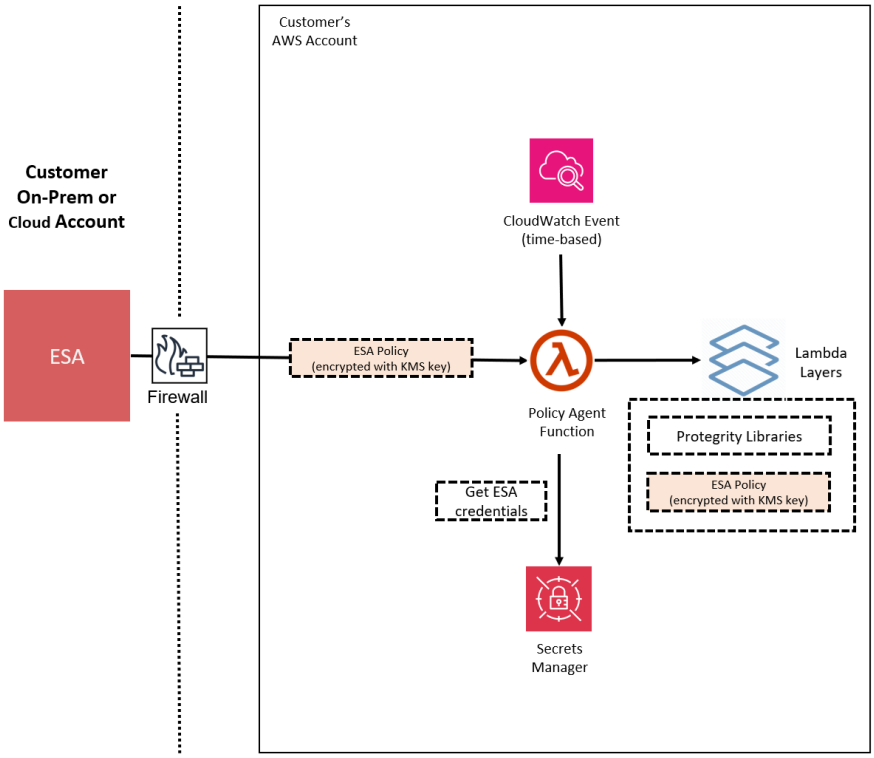


Figure 2-2: Policy Agent high-level architecture

The Protegrity Policy Agent requires network access to an Enterprise Security Administrator (ESA). Most organizations install the ESA on-premise. Therefore, it is recommended that the Policy Agent is installed into a private subnet with a Cloud VPC using a NAT Gateway to enable this communication through a corporate firewall.

The ESA is a soft appliance that must be pre-installed on a separate server. It is used to create and manage security policies.

For more information about installing the ESA, and creating and managing policies, refer the *Policy Management Guide.*

2.5 Log Forwarding Architecture

Audit logs are by default sent to CloudWatch as long as the function's execution role has the necessary permissions. The Protegrity Product can also be configured to send audit logs to ESA. Such configuration requires deploying Log Forwarder component which is available as part of Protegrity Product deployment bundle. The diagram below shows additional resources deployed with Log Forwarder component.

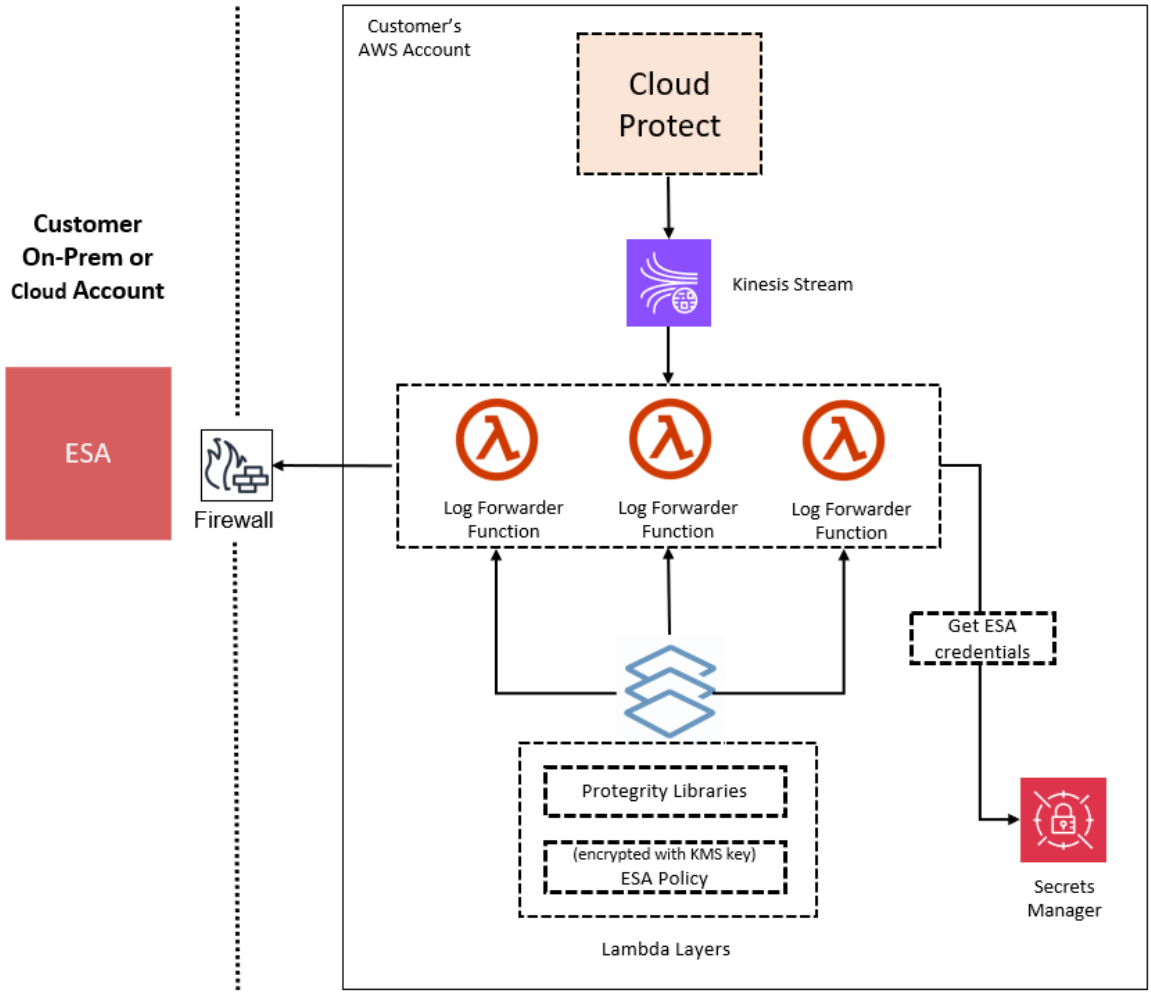


Figure 2-3: Log Forwarder Architecture

The log forwarder component includes Amazon Kinesis data stream and the forwarder Lambda function. Amazon Kinesis stream is used to batch audit records before sending them to forwarder function, where similar audit logs are aggregated before sending to ESA. Aggregation rules are described in the *Protegrity Log Forwarding guide.* When the protector function is configured to send audit logs to log forwarder, audit logs are aggregated on the protector function before sending to Amazon Kinesis. Due to specifics of the Lambda runtime lifecycle, audit logs may take up to 15 minutes before being sent to Amazon Kinesis. Protector function exposes configuration to minimize this time which is described in the protector function installation section.

The security of audit logs is ensured by using HTTPS connection on each link of the communication between protector function and ESA. Integrity and authenticity of audit logs is additionally checked on log forwarder which verifies individual logs signature. The signature verification is done upon arrival from Amazon Kinesis before applying aggregation. If signature cannot be verified, the log is forwarded as is to ESA where additional signature verification can be configured. Log forwarder function uses basic auth and optional certificate verification to authenticate calls to ESA. Basic auth credentials are stored securely in AWS Secrets Manager.

To learn more about individual audit log entry structure and purpose of audit logs, refer to *Audit Logging* section in this document. Installation instruction can be found in the *Audit Log Forwarder installation*.

The audit log forwarding requires network access from the cloud to the ESA. Most organizations install the ESA on-premise. Therefore, it is recommended that the Log Forwarder Function is installed into a private subnet with a Cloud VPC using a NAT Gateway to enable this communication through a corporate firewall.

2.6 Access Control

The following mechanisms are available for controlling and restricting access to the endpoint:

- **IAM policy:** The IAM resource policy controls which IAM users or services may invoke Protect operations. IAM policies can be applied to the API Gateway and/or the Lambda directly depending on allowable access patterns and the client.
- **JWT tokens:** The Lambda can be configured to use JSON Web Tokens (JWT) with optional verification. JSON Web Tokens (JWT) are an open, industry-standard RFC 7519 method for representing claims securely between two parties. JWT provides a mechanism for implementing custom authentication or integrating with AWS Cognito.

2.6.1 AWS Resources

Access and authorization between various AWS services involved in this architecture are achieved through IAM resource policies. For instance, the Amazon Lambda resource-based policy can restrict requests to the Amazon API Gateway or optionally allow direct invocation to the Lambda function itself. The installation steps provide a default recommended configuration. Alternative IAM role configurations are shown in the appendices in this document.

2.6.2 REST API Authentication

AWS API Gateway supports multiple mechanisms for controlling and managing access to the product.

In standard solutions, AWS API Gateway will authorize access tokens generated in the identity provider. When setting up an AWS API Gateway method to require an authorization, customers can leverage AWS Signature Version 4 or Lambda authorizers to support their organization's bearer token auth strategy.

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-to-api.html

2.6.3 REST API Authorization

Once the request is authenticated and authorized in the API Gateway, Protegrity Lambda Protector validates the user received in the authorization header of JWT, and the data element and security operations (protect or unprotect) from the payload with Protegrity Security Policy.

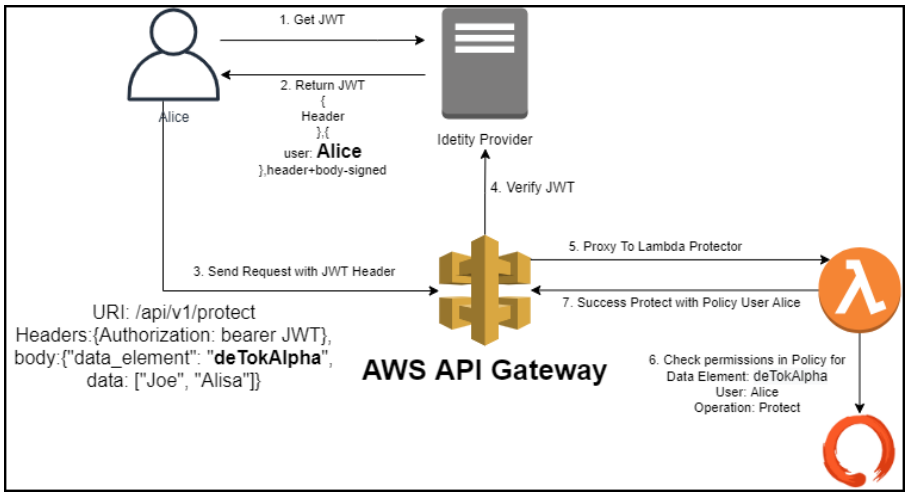


Figure 2-4: REST API authentication and authorization with identity provider

If the API Gateway is not used or configured to verify JWT tokens, the product can be configured to perform the JWT verification in the Lambda function itself.

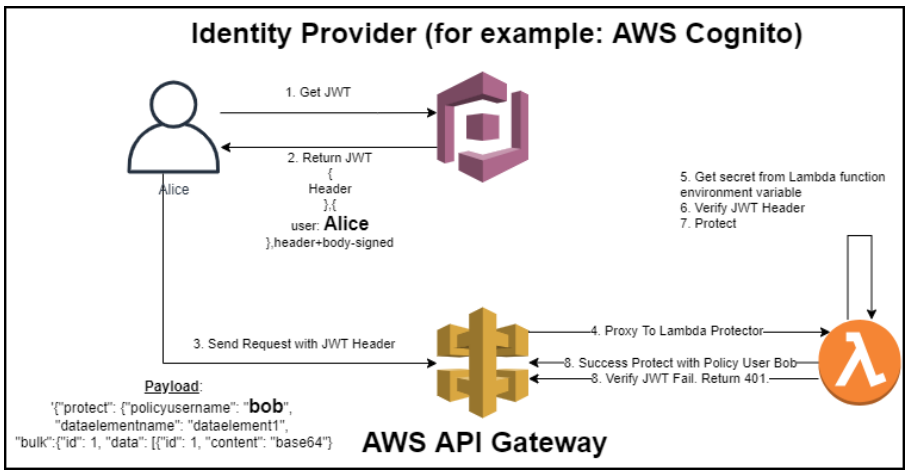


Figure 2-5: REST API authentication and authorization with Protect lambda function

Chapter 3

Installation

3.1 AWS Services

The following table describes the AWS services that may be a part of your Protegrity installation.

Service	Description
Lambda	Provides serverless compute for Protegrity protection operations and the ESA integration to fetch policy updates or deliver audit logs.
API Gateway	Provides the endpoint and access control.
KMS	Provides secrets for envelope policy encryption/decryption for Protegrity.
Secrets Manager	Provides secrets management for the ESA credentials .
S3	Intermediate storage location for the encrypted ESA policy layer.
Kinesis	Required if Log Forwarder is to be deployed. Amazon Kinesis is used to batch audit logs sent from protector function to ESA.
VPC & NAT Gateway	Optional. Provides a private subnet to communicate with an on-prem ESA.
CloudWatch	Application and audit logs, performance monitoring, and alerts. Scheduling for the policy agent.

3.2 ESA Version Requirements

The Cloud API Protector and Log Forwarder functions require a security policy from a compatible ESA version.

The table below shows compatibility between different Cloud API versions and ESA versions.

> **Note:**
> For the latest up-to-date information refer to: *Protegrity Compatibility Matrix*

Table 3-1: ESA Version Compatibility

Protector Version	ESA Version			
	8.x	**9.0**	**9.1 & 9.2**	**10.0**
2.x	No	Yes	*	No
3.0.x & 3.1.x	No	No	Yes	No
3.2.x	No	No	Yes	*

Legend	
Yes	Protector was designed to work with this ESA version
No	Protector will not work with this ESA version
*	Backward compatible policy download supported: • Data elements and features which are common between this and previous ESA versions will be downloaded • Data elements and features which are new to this ESA version and do not exist in previous ESA version will not be downloaded

3.3 Prerequisites

Requirement	Detail
Protegrity distribution and installation scripts	These artifacts are provided by Protegrity
Protegrity ESA 9.1+	The Cloud VPC must be able to obtain network access to the ESA
AWS Account	Recommend creating a new sub-account for Protegrity Serverless

3.4 Required Skills and Abilities

Role / Skillset	Description
AWS Account Administrator	To run CloudFormation (or perform steps manually), create/configure a VPC and IAM permissions.
Protegrity Administrator	The ESA credentials required to extract the policy for the Policy Agent

Role / Skillset	Description
Network Administrator	To open firewall to access ESA and evaluate AWS network setup

Tip:
During the installation you will need output of steps, such as resources names and ids, in early step. We recommend copying the following cheat sheet into a notepad and fill in the information as you progress with the installation.

AWS Account ID: _____

AWS Region (AwsRegion): _____

S3 Bucket name (ArtifactS3Bucket): _____

KMS Key ARN (AWS_KMS_KEY_ID): _____

ProtectLambdaPolicyName: _____

Role ARN (LambdaExecutionRoleArn): _____

ApiGatewayId: _____

ProtectFunctionName: _____

ProtectLayerName: _____

ESA IP address: _____

VPC name: _____

Subnet name: _____

Policy Agent Security Group Id: _____

ESA Credentials Secret Name: _____

Policy Name: _____

Agent Lambda IAM Execution Role Name: _____

3.5 Pre-Configuration

The following steps are required for installing the Protegrity product.

3.5.1 Provide AWS sub-account

Identify or create an AWS account where the Protegrity solution will be installed. It is recommended that a new AWS sub-account be created. This can provide greater security controls and help avoid conflicts with other applications that might impact regional account limits. An individual with the Cloud Administrator role will be required for some subsequent installation steps.

AWS Account ID: _____

AWS Region (AwsRegion): _____

3.5.2 Create S3 bucket for Installing Artifacts

This S3 bucket will be used for the artifacts required by the CloudFormation installation steps. This S3 bucket must be created in the region that is defined in *Provide AWS sub-account*

1. Sign in to the AWS Management Console and open the Amazon S3 console.
2. Change region to the one determined in *Provide AWS sub-account*
3. Click **Create Bucket.**
4. Enter a unique bucket name:
   For example, *protegrity-install.us-west-2.example.com*
5. Upload the installation artifacts to this bucket. Protegrity will provide the following three artifacts:
   - *protegrity-protect-<version>.zip*
   - *protegrity-agent-<version>.zip*
   - *protegrity-external-extension-<version>.zip*
   - *protegrity-sample-policy-<version>.zip*

> **Important:**
> The deployment package you receive from Protegrity must be extracted to reveal the Protegrity artifacts. CloudFormation requires them in the provided the *.zip* format. **Do not** extract the individual Protegrity artifacts. Upload these artifacts to the S3 bucket created.

S3 Bucket name (ArtifactS3Bucket): _____

3.5.3 Create KMS Key

The Amazon Key Management Service (KMS) provides the ability for the Protegrity Serverless solution to encrypt and decrypt the Protegrity Security Policy.

> **Note:**
> It is recommended to host the KMS key in a **separate** AWS sub-account. This allows **dual control**, separating the responsibility between the key administrator and the Protegrity Serverless account administrator.

To create KMS key:

1. In the AWS sub-account where the KMS key will reside, select the region.
2. Navigate to **Key Management Service** > **Create Key**.
3. Choose a **Symmetric** key type. **Key usage** should be **Encrypt and decrypt**. Click **Next**.
4. Create alias and optional description, such as, *Protegrity-Serverless* and click **Next**.
5. Define key administrative permissions, the IAM user who will administrate the key.

> **Note:**

> It is recommended the administrator be different than the administrator of the Protegrity Serverless account

6. Click **Next**.
7. Define the key usage permissions.
8. In **Other AWS accounts**, enter the AWS account id used for the Protegrity Serverless installation.
9. Continue on to create the key. If there is a concern this permission is overly broad, then you can return later to restrict access to the role of two Protegrity Serverless Lambda as principals. Click to open the key in the list and record the ARN.
   KMS Key ARN (AWS_KMS_KEY_ID): _____

3.6 Protect Service Installation

The following steps use CloudFormation to install Protegrity and the API Gateway.

3.6.1 Preparation

Ensure that all the steps in *Pre-Configuration* are performed.

1. Login to the AWS sub-account console where Protegrity will be installed.
2. Ensure that the required CloudFormation templates provided by Protegrity are available on your local computer.

3.6.2 Create Protect Lambda IAM Execution Policy

This task defines a policy used by the Protegrity Lambda function to write CloudWatch logs and access the KMS encryption key to decrypt the policy.

Perform the following steps to create the Lambda execution role and required policies.

➤ To create protect lambda IAM execution policy:

1. From the AWS IAM console, select **Policies** > **Create Policy**.
2. Select the **JSON** tab and copy the following sample policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudWatchWriteLogs",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "KmsDecrypt",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
```

```
            "arn:aws:kms:*:*:key/*"
        ]
    }
  ]
}
```

3. For the KMS policy, replace the *Resource* with the ARN for the KMS key created in a previous step.
4. Select **Next**, type in a policy name, for example, *ProtegrityProtectLambdaPolicy* and **Create Policy**. Record the policy name: **ProtectLambdaPolicyName:_____**

3.6.3 Create Protect Lambda IAM Role

The following steps create the role to utilize the policy defined in *Create Protect Lambda IAM Execution Policy*.

To create protect lambda IAM execution role:

1. From the AWS IAM console, select **Roles** > **Create Role**.
2. Select **AWS Service** > **Lambda** > **Next**.
3. In the list, search and select the policy created in *Create Protect Lambda IAM Execution Policy*.
4. Click **Next**
5. Type the role name, for example, *ProtegrityProtectRole*
6. Click **Create role**
7. Record the role ARN.
   Role ARN (LambdaExecutionRoleArn): _____

3.6.4 Install through CloudFormation

The following steps describe the deployment of the Lambda function.

1. Access CloudFormation and select the target AWS Region in the console.
2. Click **Create Stack** and choose **With new resources.**
3. Specify the template.
4. Select **Upload a template file.**
5. Upload the Protegrity-provided CloudFormation template called *pty_protect_api_cf.json* and click **Next.**
6. Specify the stack details. Enter a stack name.

> **Note:**
> The stack name will be appended to all the services created by the template.

7. Enter the required parameters. All the values were generated in the pre-configuration steps.

Table 3-2: CloudFormation template parameters

Parameter	Description	Default Value
ArtifactS3Bucket	The name of S3 bucket created during the pre-configuration steps	

Parameter	Description	Default Value
LambdaExecutionRoleArn	The ARN of Lambda role created in the prior step	
MinLogLevel	The minimum log level for the protect function. Supported values: [off, severe, warning, info, config, all]	severe
AllowAssumeUser (Optional)	If 0 is set, the user in the request body will be ignored and the REST API authenticated user will be the acting user. Supported values: [0, 1]	0
LambdaAuthorization (Optional)	If "jwt" is set, then Authorization header for JWT will be required in the AWS Protect Lambda request. Any other value is ignored and effective policy user is taken from request payload. Supported values: ["jwt", ""]	""
jwtUsernameClaim (Optional)	The JWT claim with username. Common claims: name, preferred_username, cognito:username Also accepts ordered list of claim names in JSON array format, e.g. ["username", "email"]	cognito:username
jwtVerify (Optional)	If 1 is set, then **jwtSecretBase64** is required. Only applicable when **LambdaAuthorization** is set to "jwt". Supported JWT algorithms are: RS256, RS384, RS512. While algorithms HS256, HS384, HS512 are supported, they are not recommended for use.	0
jwtSecretBase64 (Optional)	Required when **jwtVerify** is set to 1 and **Authorization** is set to "jwt". The secret must be provided in base64 encoding. It is recommended to only use public key (asymmetric algorithm).	""

8. The log forwarder parameters can be provided later after log forwarder is deployed. If you are not planning to deploy log forwarder you can skip this step.

Table 3-3: Log Forwarder Parameters

Parameter	Description
KinesisLogStreamArn	The ARN of the AWS Kinesis stream where audit logs will be sent for aggregation

Parameter	Description
AuditLogFlushInterval	Time interval in seconds used to accumulate audit logs before sending to Kinesis. Default value: 30. See *Log Forwarder Performance* section for more details.

9. Click **Next** with defaults to complete CloudFormation.
10. After CloudFormation is completed, select the **Outputs** tab in the stack.
11. Record the following values:

    ApiGatewayId: _____

    ProtectFunctionName: _____

    ProtectFunctionProductionAlias: _____

    ProtectLayerName: _____

3.6.5 Test Connectivity

Perform the following steps to verify if the API Gateway is working correctly with the Protegrity product.

1. 1. In the AWS console Access **API Gateway.**
2. Search for API (CloudFormation output ApiGatewayId value).
3. Select **Resources** > **/v1/unprotect POST.**
4. Click on **Test** tab.
5. Provide the following input in **Headers.**

```
Authorization:
```

6. Provide the following input in **Request body**.

```
{"user": "test_user", "data_element": "alpha", "data": ["UtfVk UHgcD!"]}
```

7. Click **Test.**
8. Verify the response **Status** is 200 and the **Response Body** is as follows:

```
{"encoding": "utf8", "results": ["hello world!"], "success": true}
```

   For example:

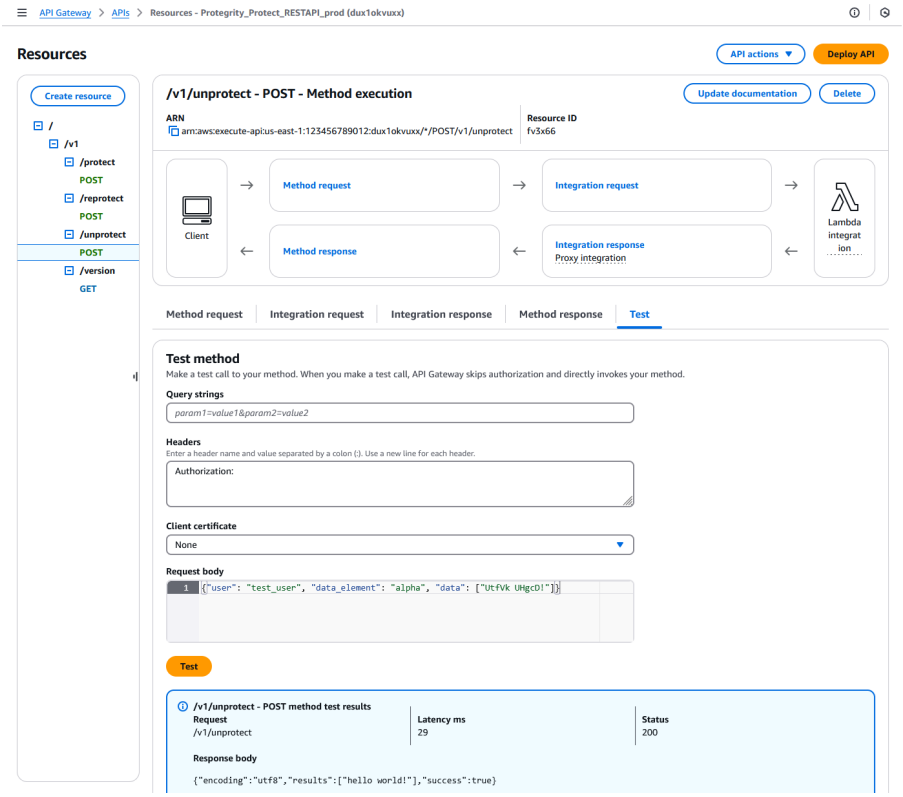


Figure 3-1: Test connectivity

3.6.6 Troubleshooting

Table 3-4: Troubleshooting errors

Error	Action
5xx error	1. Try running the Lambda directly. Open the Lambda function and create the following test case: `{ "body": "{\"user\":\"test_user\",\"data_element\": \"alpha\", \"data\": [\"UtfVk UHgcD!\"]}", "path": "/unprotect", "headers": { "Authorization": "" } }` If this step fails, then check the console for the meaningful error.

3.6.7 Setting up Authentication

This step describes how to setup an AWS API Gateway Authorization.

By default, the API Gateway is configured to use AWS_IAM Authorization.

1. From AWS console access API Gateway.
2. If you are using **AWS API Gateway Authorizer,** ensure that the Authorizer is configured in **Authorizers.**
3. Go to **Resources** > **/v1/protect POST** > **Method Request tab** > **Click Edit button**

4. Select from **Authorization.** dropdown.
5. Click **Save** button.
6. Click **Deploy API** button to deploy to **pty** stage.

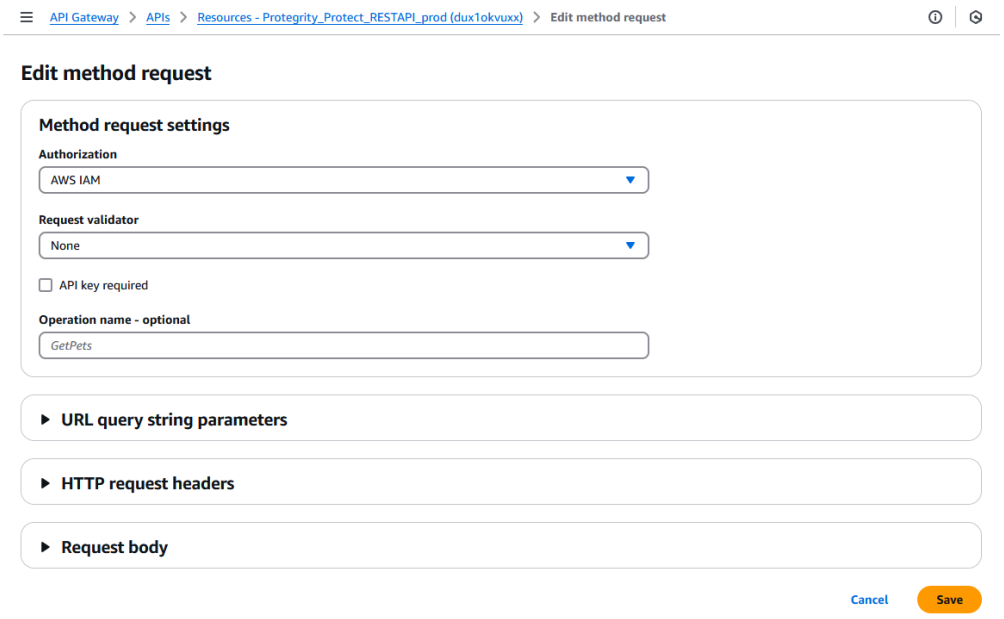


Figure 3-2: Setting up Authentication

3.6.8 Protect Lambda Configuration

After Cloud Formation stack is deployed, Protector Lambda can be reconfigured based on the authentication selected in previous stage. From your AWS console navigate to lambda and select following lambda: Protegrity_Protect_RESTAPI_<STACK_NAME>. Scroll down to Environment variables section, select Edit and replace the entries based on the following information.

Table 3-5: Protect Lambda configuration using environment variables

Environment Variable	Description	Notes
authorization	If "jwt" is set, Authorization header with JWT will be required in the AWS Protect Lambda request. Any other value is ignored and effective policy user is taken from request payload. Supported Values: ["jwt", ""]	If "jwt" is set, any request without valid JWT in the Authorization header, will result in error from API Gateway: 502 Bad Gateway. Default Value: ""
allow_assume_user	If 0 is set, API Gateway user will not be used, and the effective user is the JWT user. Supported Values: [0, 1]	Applicable when **authorization** is set to "jwt". Default Value: 0
jwt_user_claim	The JWT claim with username. Common claims: name, preferred_username, cognito:username	Applicable when **authorization** is set to "jwt" Default Value: cognito:username
jwt_verify	If 1 is set, **jwt_secret_base64** is required. Only applicable when **authorization** is set to	Applicable when **authorization** is set to "jwt"

Environment Variable	Description	Notes
	"jwt". Supported JWT algorithm RS256, RS384, RS512. While algorithms HS256, HS384, HS512 are supported, they are not recommended for use.	
jwt_secret_base64	Required when **jwt_verify** is set to 1 and **Authorization** is set to "jwt". The secret must be provided in base64 encoding. It is recommended to only use public key (asymmetric algorithm).	Applicable when **authorization** is set to "jwt" and **jwt_verify** is set to 1
service_user	If set, it will be used as effective policy user.	**service_user** should be used when the Cloud API should always run as single service user or account no matter what user is in the request. **service_user** will always take priority over users in the payload and in the JWT header.
USERNAME_REGEX	If set, the effective policy user will be extracted from the user in the request using supplied regular expression.	**Note:** See *Configuring Regular Expression to Extract Policy Username* to learn how to extract username from the request

3.7 Policy Agent Installation

The following sections will install the Policy Agent. The Policy Agent polls the ESA and deploys the policy to Protegrity Serverless as a static resource. Some of the installation steps are not required for the operation of the software but recommended for establishing a secure environment. Contact Protegrity Professional Services for further guidance on configuration alternatives in the Cloud.

3.7.1 ESA Server

Policy Agent Lambda requires ESA server running and accessible on TCP port 8443.

Note down ESA IP address:

ESA IP Address (EsaIpAddress): _____

3.7.2 Certificates on ESA

Whether your ESA is configured with default self-signed certificate or your corporate CA certificate, Policy Agent can validate authenticity of ESA connection using CA certificate. The process for both scenarios is the same:

- Obtain CA certificate
- Convert CA certificate to a value accepted by Policy Agent
- Provide converted CA certificate value to Policy Agent

> **Fastpath:** If ESA is configured with publicly signed certificates, this section can be skipped, since the Policy Agent will use public CA to validate ESA connection.

To obtain self-signed CA certificate from ESA:

1. Log in to ESA Web UI.
2. Select **Settings** > **Network** > **Manage Certificates**.
3. Hover over **Server Certificate** and click on download icon to download the CA certificate.
4. To convert downloaded CA certificate to a value accepted by Policy Agent, open the downloaded PEM file in text editor and replace all new lines with escaped new line: \n.

   To escape new lines from command line, use one of the following commands depending on your operating system:

   Linux Bash:

   ```
   awk 'NF {printf "%s\\n",$0;}' ProtegrityCA.pem > output.txt
   ```

   Windows PowerShell:

   ```
   (Get-Content '.\ProtegrityCA.pem') -join '\n' | Set-Content 'output.txt'
   ```

5. Record the certificate content with new lines escaped.

   ESA CA Server Certificate (EsaCaCert): _____

   This value will be used to set PTY_ESA_CA_SERVER_CERT Lambda variable in section *Policy Agent Lambda Configuration*

For more information about ESA certificate management refer to *Certificate Management Guide* in ESA documentation.

3.7.3 Identify or Create a New VPC

Establish a VPC where the Policy Agent will be hosted. This VPC will need connectivity to the ESA. The VPC should be in the same account and region established in *Pre-Configuration*.

VPC name: _____

3.7.4 Create a Private Subnet

If a new VPC was created, then create a private subnet within the VPC. This ensures that the Policy Agent will not be visible on the Internet.

Subnet name: _____

3.7.5 Create an Internet Gateway, NAT Gateway, and Routing

If a new VPC was created, then add an Internet Gateway, then add a NAT gateway and an elastic IP address. Policy Agent will use these components to access an on-prem ESA. A Routing Table and Network ACL may need to be configured for outbound access to the ESA.

Elastic IP: _____

3.7.6 Create a VPC Endpoint

If an internal VPC was created, then add VPC Endpoints, which will be used by the Policy Agent to access AWS services. Policy Agent needs access to the following AWS services:

Table 3-6: VPC Endpoints

Type	Service name
Interace	*com.amazonaws.{REGION}.secretsmanager*
Interace	*com.amazonaws.{REGION}.kms*
Gateway	*com.amazonaws.{REGION}.s3*
Interace	*com.amazonaws.{REGION}.lambda*

3.7.7 Identify or Create Security Groups

Policy Agent and cloud-based ESA appliance use AWS security groups to control traffic that is allowed to leave and reach them. Policy Agent runs on schedule and is mostly concerned with allowing traffic out of itself to ESA and AWS services it depends on. ESA runs most of the time and it must allow Policy Agent to connect to it.

Policy Agent security group must allow outbound traffic using rules described in the table below. To edit security group navigate:

From VPC > Security Groups > Policy Agent Security Group configuration

Table 3-7: Outbound traffic rules of Policy Agent security group

Type	Protocol	Port Range	Destination	Reason
Custom TCP	TCP	8443	Policy Agent Lambda SG	ESA communication
HTTPS	TCP	443	Any	AWS services

Record Policy Agent security group ID:

Policy Agent Security Group Id: _____

Policy Agent will reach out to ESA on port 8443. Create following inbound security group rule for cloud-based ESA appliance to allow connections from Policy Agent:

Table 3-8: Inbound rules of ESA security group to allow inbound traffic from Policy Agent

Type	Protocol	Port Range	Source
Custom TCP	TCP	8443	Policy Agent Lambda SG

3.7.8 Creating ESA Credentials

Policy Agent Lambda requires ESA credentials to be provided as one of the three options.

> **Note:**
> The username and password of the ESA user requires role with *DPS Admin* and *Export Certificates* permissions. *Security Administrator* is one of the predefined roles which contains the above permissions, however for separation of duties it is recommended to create custom role.

3.7.8.1 Option 1: Secrets Manager

Creating secrets manager secret with ESA username and password.

1. From the AWS Secrets Manager Console, select **Store New Secret**.
2. Select **Other Type of Secrets**.
3. Specify the **username** and **password** key value pair.
4. Select the encryption key or leave default AWS managed key. Click Next.
5. Specify the Secret Name and record it.
   ESA Credentials Secret Name: _____

3.7.8.2 Option 2: KMS Encrypted Password

ESA password is encrypted with AWS KMS symmetric key.

1. Create AWS KMS symmetric key which will be used to encrypt ESA password. See *Create KMS Key* for instructions on how to create KMS symmetric key using AWS console.
2. Record KMS Key ARN.
   ESA PASSWORD KMS KEY ARN: _____
3. Run AWS CLI command to encrypt ESA password. Below you can find sample Linux aws cli command. Replace **<key_arn>** with KMS symmetric key ARN.

   ```
   aws kms encrypt --key-id <key_arn> --plaintext $(echo '<esa_password>' | base64 )
   ```

4. Sample output.

   ```
   {
     "CiphertextBlob": "esa_encrypted_password",
     "KeyId": "arn:aws:kms:region:aws_account:key/key_id ",
     "EncryptionAlgorithm": "SYMMETRIC_DEFAULT"
   }
   ```

5. Record ESA username and encrypted password.
   ESA USERNAME: _____
   ESA ENCRYPTED PASSWORD: _____

3.7.8.3 Option 3: Custom AWS Lambda function

With this option ESA username and password are returned by a custom AWS Lambda function. This method may be used to get the username and password from external vaults.

1. Create AWS Lambda in any AWS supported runtime.
   a. There is no input needed.
   b. The Lambda function must return the following response schema.

```
response:
type: object
  properties:
    username: string
    password: string
```

   For example,

```
example output: {"username": "admin", "password": "Password1234"}
```

   c. Sample AWS Lambda function in Python:

```
import json

def lambda_handler(event, context):

    return {"username": "admin", "password": "password1234"}
```

> **Warning:** Protegrity does not recommend hardcoding ESA username and password in the clear.

2. Record the Lambda name:
   Custom AWS lambda for ESA credentials: _____

3.7.9 Create Agent Lambda IAM policy

Follow the steps below to create Lambda execution policies.

Create Agent Lambda IAM policy

1. From AWS IAM console, select **Policies** > **Create Policy.**
2. Select **JSON** tab and copy the following snippet.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EC2ModifyNetworkInterfaces",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CloudWatchWriteLogs",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "LambdaUpdateFunction",
      "Effect": "Allow",
      "Action": [
        "lambda:UpdateFunctionConfiguration"
```

```
      ],
      "Resource": [
        "arn:aws:lambda:*:*:function:*"
      ]
    },
    {
      "Sid": "LambdaReadLayerVersion",
      "Effect": "Allow",
      "Action": [
        "lambda:GetLayerVersion",
        "lambda:ListLayerVersions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "LambdaDeleteLayerVersion",
      "Effect": "Allow",
      "Action": "lambda:DeleteLayerVersion",
      "Resource": "arn:aws:lambda:*:*:layer:*:*"
    },
    {
      "Sid": "LambdaPublishLayerVersion",
      "Effect": "Allow",
      "Action": "lambda:PublishLayerVersion",
      "Resource": "arn:aws:lambda:*:*:layer:*"
    },
    {
      "Sid": "S3GetObject",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::*/*"
    },
    {
      "Sid": "S3PutObject",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::*/*"
    },
    {
      "Sid": "KmsEncrypt",
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:*:*:key/*"
      ]
    },
    {
      "Sid": "SecretsManagerGetSecret",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:*:*:secret:*"
      ]
    },
    {
      "Sid": "LambdaGetConfiguration",
      "Effect": "Allow",
      "Action": [
          "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
          "arn:aws:lambda:*:*:function:*"
      ]
    }
```

```
  ]
}
```

3. Replace wildcard * with the region, account, and resource name information where required.
4. This step is required if KMS is used to encrypt ESA password.

   Add policy entry below. Replace **ESA PASSWORD KMS KEY ARN** with the value recorded in *Option 2: KMS Encrypted Password*.

```
{
  "Sid": "KmsDecryptEsaPassword",
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt"
  ],
  "Resource": [
    "ESA PASSWORD KMS KEY ARN"
  ]
}
```

5. Select **Next** type in the policy name and **Create Policy**. Record policy name:

   Policy Name: _____

3.7.10 Create Agent Lambda IAM Role

Perform the following steps to create Agent Lambda execution IAM role.

➤ To create agent Lambda IAM role:

1. From AWS IAM console, select **Roles** > **Create Role**.
2. Select **AWS Service** > **Lambda** > **Next**.
3. Select the policy created in *Create Agent Lambda IAM policy*.
4. Proceed to **Name, Review and Create**.
5. Type the role name, for example, *ProtegrityAgentRole* and click **Confirm**.
6. Select **Create role**.
7. Record the role ARN.

   Agent Lambda IAM Execution Role Name: _____

3.7.11 Corporate Firewall Configuration

If an on-premise firewall is used, then the firewall must allow access from the NAT Gateway to an ESA. The firewall must allow access from the NAT Gateway IP to ESA via port 8443 and 443.

3.7.12 CloudFormation Installation

Create the Policy Agent in the VPC using the CloudFormation script provided by Protegrity.

1. Access the CloudFormation service.

2. Select the target installation region.
3. Create a stack with new resources.
4. Upload the Policy Agent CloudFormation template *(file name: pty_agent_cf.json).*
5. Specify the following parameters for Cloud Formation:

Parameter	Description	Note
VPC	VPC where the Policy Agent will be hosted	*Identify or Create a New VPC*
Subnet	Subnet where the Policy Agent will be hosted	*Create a Private Subnet*
PolicyAgentSecurityGroupId	Security Group Id, which allows communication between the Policy Agent and the ESA	*Identify or Create Security Groups*
LambdaExecutionRoleArn	Agent Lambda IAM execution role ARN allowing access to the S3 bucket, KMS encryption Key, Lambda and Lambda Layer	*Create Agent Lambda IAM Role*
ArtifactS3Bucket	S3 bucket name with deployment package for the Policy Agent	*Create S3 bucket for Installing Artifacts*
CreateCRONJob	Set to True to create a CloudWatch schedule for the agent to run.	Default: False

3.7.13 Policy Agent Lambda Configuration

After the CloudFormation stack is deployed, the Policy Agent Lambda must be configured with parameters recorded in earlier steps. From your AWS Console, navigate to lambda and select the following Lambda.

Protegrity_Agent_<STACK_NAME>

Select **Configuration** tab and scroll down to the **Environment variables** section. Select **Edit** and replace all entries with the actual values.

Parameter	Description	Notes
PTY_ESA_IP	ESA IP address or hostname	*ESA Server*
PTY_ESA_CA_SERVER_CERT	ESA self-signed CA certificate or your corporate CA certificate used by policy Agent Lambda to ensure ESA is the trusted server.	Recorded in step *Certificates on ESA* In case ESA is configured with publicly signed certificates, the PTY_ESA_CA_SERVER_CERT configuration will be ignored.
PTY_ESA_CREDENTIALS_SECRET	ESA username and password (encrypted value by AWS Secrets Manager)	*Option 1: Secrets Manager*

Parameter	Description	Notes
AWS_KMS_KEY_ID	KMS key id or full ARN e.g. arn:aws:kms:us-west-2:112233445566:key/bfb6c4fb-509a-43ac-b0aa-82f1ca0b52d3	*Create KMS Key*
AWS_POLICY_S3_BUCKET	S3 bucket where the encrypted policy will be written	S3 bucket of your choice
AWS_POLICY_S3_FILENAME	Filename of the encrypted policy stored in S3 bucket	Default: *protegrity-policy.zip*
AWS_PROTECT_FN_NAME	Comma separated list of Protect function names or ARNs	Step 9 *Install through CloudFormation*
DISABLE_DEPLOY	This flag can be either 1 or 0. If set to 1, then the agent will not update PTY_PROTECT lambda with the newest policy. Else, the policy will be saved in the S3 bucket and deployed to the Lambda Layer	Default: 0
AWS_POLICY_LAYER_NAME	Lambda layer used to store the Protegrity policy used by the PTY_PROTECT function	
POLICY_LAYER_RETAIN	Number of policy versions to retain as backup. (e.g. 2 will retain the latest 2 policies and remove older ones). -1 retains all.	Default: 2
POLICY_PULL_TIMEOUT	Time in seconds to wait for the ESA to send the full policy	Default: 20s
ESA_CONNECTION_TIMEOUT	Time in seconds to wait for the ESA response	Default: *5s*
LOG_LEVEL	Application and audit logs verbiage level	Default: INFO Allowed values: DEBUG – the most verbose INFO, WARNING, ERROR – the least verbose
PEP_CONFIG_EMPTY_STRING	Override default behavior. Empty string response values are returned as null values. For instance, (un)protect('') -> null (un)protect('') -> ''	Default: empty Allowed values: null empty
PEP_CONFIG_CASE_SENSITIVE	Specifies whether policy usernames should be case sensitive	Default: no Allowed values: yes no

Parameter	Description	Notes
PTY_ADDIPADDRESSHEADER	When enabled, agent will send its source IP address in the request header. This configuration works in conjunction with ESA hubcontroller configuration ASSIGN_DATASTORE_USING_NODE_IP (default=false). See *Associating ESA Data Store With Cloud Protect Agent* for more information.	Default: yes Allowed values: yes no
PTY_ESA_USERNAME	Plaintext ESA username which is used together with PTY_ESA_ENCRYPTED_PASSWORD as an optional ESA credentials	*Option 2: KMS Encrypted Password* Presence of this parameter will cause PTY_ESA_CREDENTIALS_SECRET to be ignored
PTY_ESA_ENCRYPTED_PASSWORD	ESA password encrypted with KMS symmetric key. Example AWS cli command to generate the value: aws kms encrypt --key-id <your key ARN> --plaintext '<your-esa-password-base64> '	*Option 2: KMS Encrypted Password* Presence of this parameter will cause PTY_ESA_CREDENTIALS_SECRET to be ignored Value must be base64 encoded
EMPTY_POLICY_S3	This flag can be either 1 or 0. If set to 1, then the agent will remove the content of the policy file in S3 bucket, but will keep the checksum in the metadata. Else, the policy will be saved in the S3 bucket and not removed.	Default: 0
PTY_ESA_CREDENTIALS_LAMBDA	Lambda function to return ESA credentials	Recorded in step *Option 3: Custom AWS Lambda function* LAMBDA FOR ESA CREDENTIALS. Presence of PTY_ESA_USERNAME, or PTY_ESA_CREDENTIALS_SECRET will cause this value to be ignored. The Policy Agent Lambda must have network access and IAM permissions to invoke the custom ESA Credentials Lambda you have created in *Option 3: Custom AWS Lambda function*.

3.7.14 Test Installation

Open the Lambda and configure **Test** to execute the lambda and specify the default test event. Wait for around 20 seconds for the Lambda to complete. If policy is downloaded successfully, then a success message appears.

Navigate to the *AWS_POLICY_S3_BUCKET* bucket and verify that the *AWS_POLICY_S3_FILENAME* file was created.

3.7.15 Troubleshooting

Lambda Error	Example Error	Action
Task timed out after x seconds ESA connection error. Failed to download certificates	`2020-10-06T23:40:54.121Z 2dc84942-b5cc-4be9-aa4c-965f322307e4 Task timed out after 90.09 seconds`	1. Ensure that there is network connectivity between the Lambda and ESA. Check the Security groups and/or Network firewall configuration 2. When using internal VPC, AWS Lambda needs to have access to AWS Network. The Policy Agent Lambda can start using Secrets Manager with Amazon VPC endpoints by creating an Amazon VPC endpoint for Secrets Manager.
Policy Pull takes a long time	`{ "errorMessage": "Timeout! Unable to download policy in 20 seconds.", "errorType": "Exception", "stackTrace": [...] }`	1. Increase POLICY_PULL_TIMEOUT. 2. Ensure that there is at least 1 policy with datastore matching the Lambda Policy Agent. Other considerations: a. Policy has default datastore. b. Policy has datastores matching AWS lambda IP range (check the subnet IP Range). c. Lambda function has static IP, and at least one Data store has matching IP.
ESA connection error. Failed to download certificates. HTTP response code: 401	`{ "errorMessage": "ESA connection error. Failed to download certificates. HTTP response code: 401.", "errorType": "ConnectionError", "stackTrace": [...] }`	Ensure that the PTY_ESA_CREDENTIALS_SECRET has correct ESA username and password
An error occurred (AccessDeniedException) when calling xyz operation	`xyz Access Denied: Exception Traceback (most recent call last): … Exception: xyz Access Denied`	Ensure that the Lambda execution role has permission to call the **xyz** operation
Access Denied to Secret Manager.	`Secrets Manager Access Denied: Exception Traceback (most recent call last): … Exception: Secrets Manager Access Denied`	1. Ensure that the Lambda execution role has permissions to get the Secret Manager secret name. 2. Ensure that the Lambda execution role has permission to get the Secret Manager secret Encryption Key.
Master Key xyz unable to generate data key		Ensure that the Lambda can access **xyz** CMK key

Lambda Error	Example Error	Action

The S3 bucket **server-side encryption** is enabled, the encryption key type is **SSE-KMS** but the Policy Agent execution IAM role doesn't have permissions to encrypt using the KMS key .	`[ERROR] PolicyAgentException: An error occurred (AccessDenied) when calling the PutObject operation: Access Denied`	Add the following permissions to the Policy Agent excution role.
		`kms:Decrypt`
		`kms:GenerateDatakey`
		
		Note: When the KMS key and the Policy Agent Lambda are in separate accounts, update both the AWS KMS key and the Policy Agent execution role.

The S3 bucket has bucket policy to only allow access from within the VPC.	`An error occurred (AccessDeniedException) when calling the PublishLayerVersion operation: Your access has been denied by S3, please make sure your request credentials have permission to GetObject for BUCKET_NAME/FILENAME. S3 Error Code: AccessDenied. S3 Error Message: Access Denied`	The Policy Agent publishes a new Lambda Layer version, and the Lambda Layer service uploads the policy file from the s3 bucket and the upload request is originated from the AWS service outside the Policy Agent Lambda VPC. Update the S3 bucket resource policy to allow access from AWS Service. Sample security policy to lock down access to the vpc: `{ "Version": "2012-10-17", "Statement": [{ "Sid": "VpcRestrictions", "Effect": "Deny", "Principal": { "AWS": "*" }, "Action": "s3:*Object", "Resource": ["arn:aws:s3:::<s3_bucket_name>/*", "arn:aws:s3:::<s3_bucket_name>"], "Condition": { "Bool": { "aws:ViaAWSService": "false" }, "StringNotEquals": { "aws:sourceVpc": "<vpc_id>" } } }] }`

3.7.16 Additional Configuration

Strengthen the KMS IAM policy by granting access only to the Lambdas.

Finalize the IAM policy for the *Lambda Execution Role*. Ensure to replace wildcard * with the region, account, and resource name information where required.

For example,

```
"arn:aws:lambda:*:*:function:*" -> "arn:aws:lambda:us-east-1:account:function:function_name"
```

3.7.17 Policy Agent Schedule

If specified in *CloudFormation Installation*, the agent installation created a CloudWatch event rule, which checks for policy update on an hourly schedule. This schedule can be altered to the required frequency.

Under **CloudWatch** > **Events** > **Rules**, find *Protegrity_Agent_{stack_name}*. Click **Action** > **Edit**Set the cron expression. A cron expression can easily be defined using CronMaker, a free online tool. Refer to *http://www.cronmaker.com*.

3.8 Audit Log Forwarder Installation

The following sections provide installation steps for the Protegrity Audit Log Forwarder component in AWS. The Log Forwarder deployment allows for the audit logs generated by Protector to be delivered to ESA for auditing and governance purposes. Log Forwarder component is optional and is not required for the Protector Service to work properly. See *Log Forwarding Architecture* for more information. Some of the installation steps are not required for the operation of the software but recommended for establishing a secure environment. Contact Protegrity for further guidance on configuration alternatives in the cloud.

> **Note:** The installation steps below assume that the Log Forwarder is going to be installed in the same AWS account as the corresponding Protect Lambda Function. For instructions on how to install Log Forwarder in the AWS account separate than the Protect Lambda, please contact Protegrity.

3.8.1 ESA Audit Store Configuration

ESA server is required as the recipient of audit logs. Verify the information below to ensure ESA is accessible and configured properly.

1. ESA server running and accessible on TCP port 9200.
2. Audit Store service is configured and running on ESA. For information related to ESA Audit Store configuration, refer to *Audit Store Guide*.

3.8.2 Certificates on ESA

By default, ESA is configured with self-signed certificates, which can only be validated using self-signed CA certificate supplied in Log Forwarder configuration.

> **Note:**
> Certificate Validation can be bypassed for testing purposes, see section: *Install through CloudFormation*

Before you begin
In case ESA is configured with publicly signed certificates, this section can be skipped since the forwarder Lambda will use public CA to validate ESA certificates.

To obtain self-signed CA certificate from ESA:

1. Log in to ESA Web UI.
2. Select **Settings** > **Network** > **Manage Certificates**.
3. Hover over **Server Certificate** and click on download icon to download the CA certificate.
4. After certificate is downloaded, open the PEM file in text editor and replace all new lines with escaped new line: \n.
   To escape new lines from command line, use one of the following commands depending on your operating system:
   Linux Bash:

   ```
   awk 'NF {printf "%s\\n",$0;}' ProtegrityCA.pem > output.txt
   ```

   Windows PowerShell:

   ```
   (Get-Content '.\ProtegrityCA.pem') -join '\n' | Set-Content 'output.txt'
   ```

5. Record the certificate content with new lines escaped.
   ESA CA Server Certificate (EsaCaCert): _____
   This value will be used to set PtyEsaCaServerCert cloudformation parameter in section *Install through CloudFormation*

For more information about ESA certificate management refer to *Certificate Management Guide* in ESA documentation.

3.8.3 AWS VPC Configuration

Log forwarder Lambda function requires network connectivity to ESA, similar to Policy Agent Lambda function. Therefore, it can be hosted in the same VPC as Policy Agent.

Separate VPC can be used, as long as it provides network connectivity to ESA.

> **Note:** AWS Lambda service uses permissions in log forwarder function execution role to create and manage network interfaces. Lambda creates a Hyperplane ENI and reuses it for other VPC-enabled functions in your account that use the same subnet and security group combination. Each Hyperplane ENI can handle thousands of connections/ports as the Lambda function scales up. If more connections are needed AWS Lambda service creates additional Hyperplane ENIs. There's no additional charge for using a VPC or a Hyperplane ENI. Refer to AWS official *Lambda Hyperplane ENIs* docs for more information.

VPC Name: _____

3.8.4 VPC Subnet Configuration

Log Forwarder can be connected to the same subnet as Policy Agent or separate one as long as it provides connectivity to ESA.

Subnet Name: _____

3.8.5 NAT Gateway For ESA Hosted Outside AWS Network.

If ESA server is hosted outside of the AWS Cloud network, the VPC configured for Lambda function must ensure additional network configuration is available to allow connectivity with ESA. For instance if ESA has a public IP, the Lambda function VPC

must have public subnet with a NAT server to allow routing traffic outside of the AWS network. A Routing Table and Network ACL may need to be configured for outbound access to the ESA as well.

3.8.6 VPC Endpoint Configuration

Log Forwarder Lambda function requires connectivity to Secrets Manager AWS service. If the VPC identified in the steps before has no connectivity to public internet through the NAT Gateway, then the following service endpoint must be configured:

- *com.amazonaws.{REGION}.cloudwatch*
- *com.amazonaws.{REGION}.secretsmanager*
- *com.amazonaws.{REGION}.kms*

3.8.7 Security Group Configuration

Security groups restrict communication between Log Forwarder Lambda function and the ESA appliance. The following rules must be in place for ESA and Log Forwarder Lambda function.

From **VPC** > **Security Groups** > **Log Forwarder Security Group configuration**.

Table 3-9: Outbound Rules

Type	Protocol	Port Range	Destination	Reason
Custom TCP	TCP	9200	Log Forwarder Lambda SG	ESA Communication

Record the name of Log Forwarder security group name.

Log Forwarder Security Group Id: _____

The following port must be open for the ESA. If the ESA is running in the Cloud, then create the following security.

> **Note:** If an on-premise firewall is used, then the firewall must allow access from the NAT Gateway to an ESA. The firewall must allow access the NAT Gateway IP access to ESA via port 9200.

ESA Security Group configuration

Table 3-10: Inbound Rules

Type	Protocol	Port Range	Source
Custom TCP	TCP	9200	Log Forwarder Lambda SG

3.8.8 Configure ESA Audit Store Credentials

Log Forwarder must authenticate to ESA. There are two options for Log Forwarder to authenticate and only one of them must be configured at a time:

- [Deprecated*] Basic authentication with username and password
- Certificate authentication with client certificate and certificate key

This section provides two sets of instructions on how to configure ESA Audit Store credentials for Log Forwarder Lambda.

> **Note:** *Basic Authentication method will be removed with Cloud Protect version 4.

3.8.8.1 Basic Authentication

Audit Log Forwarder Function uses AWS Secrets Manager to store ESA credentials used during authentication.

> **Warning:** Basic Authentication method is not available on ESA 10 and higher.

Before you begin
For information on how to configure basic authentication for Audit Store on ESA refer to *Audit Store Guide*.

➤ To create secret with ESA username and password in AWS Secrets Manager.

1. From the AWS Secrets Manager Console, select **Store New Secret**.
2. Select **Other Type of Secrets**.
3. Specify the **username** and **password** key value pair.
4. Select the encryption key or leave default AWS managed key. Click Next.
5. Specify the Secret Name and record it.
   ESA Log Forwarder Credentials Secret Name: _____
   This value will be used to set EsaCredentialsSecretId cloudformation parameter in section *Install through CloudFormation*

3.8.8.2 Certificate-Based Authentication

Audit Log Forwarder Function uses AWS Secrets Manager to store ESA credentials used during authentication.

Before you begin

Download the following certificates from the */etc/ksa/certificates/plug* directory of the ESA:

- *client.key*
- *client.pem*

After certificates are downloaded, open each PEM file in text editor and replace all new lines with escaped new line: \n. To escape new lines from command line, use one of the commands below depending on your operating system.

Linux Bash:

```
awk 'NF {printf "%s\\n",$0;}' client.key > private_key.txt
awk 'NF {printf "%s\\n",$0;}' client.pem > public_key.txt
```

Windows PowerShell:

```
(Get-Content '.\client.key') -join '\n' | Set-Content 'private_key.pem'
(Get-Content '.\client.pem') -join '\n' | Set-Content 'public_key.pem'
```

For more information on how to configure client certificate authentication for Audit Store on ESA refer to *Audit Store Guide*.

➤ To create secret with ESA client certificate/key pair in AWS Secrets Manager.

1. From the AWS Secrets Manager Console, select **Store New Secret**.
2. Select **Other Type of Secrets**.
3. Specify the **private_key** and **public_key** value pair.

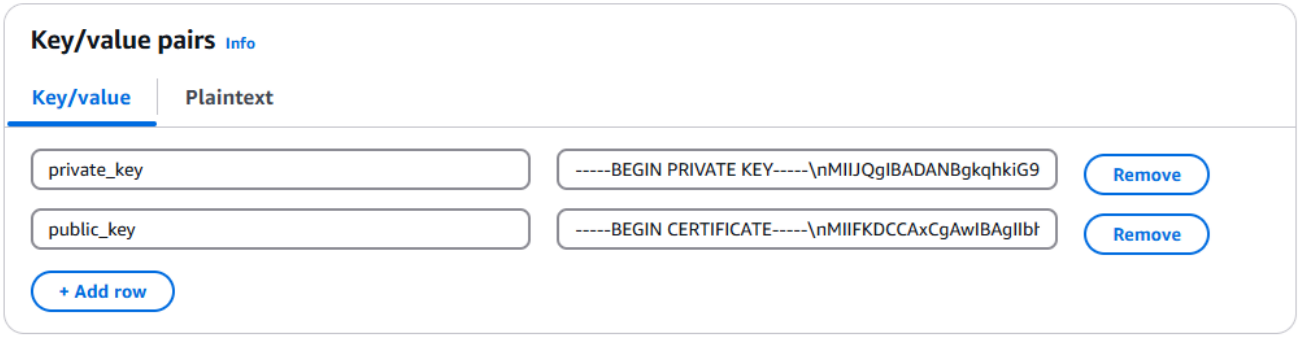


Figure 3-3: Secrets Manager

4. Select the encryption key or leave default AWS managed key.
5. Specify the Secret Name and record it below.

   ESA Client Certificate/Key Pair Secret Name: _____

   This value will be used to set PtyEsaClientCertificatesSecretId cloudformation parameter in section *Install through CloudFormation*

3.8.9 Create Audit Log Forwarder IAM Execution Policy

This task defines a policy used by the Protegrity Log Forwarder Lambda function to write CloudWatch logs, access the KMS encryption key to decrypt the policy and access Secrets Manager for log forwarder user credentials.

Perform the following steps to create the Lambda execution role and required policies:

1. From the AWS IAM console, select **Policies** > **Create Policy**.
2. Select the **JSON** tab and copy the following sample policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EC2ModifyNetworkInterfaces",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CloudWatchWriteLogs",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "*"
    },
    {
      "Sid": "KmsDecrypt",
      "Effect": "Allow",
```

```
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "arn:aws:kms:*:*:key/*"
      ]
    },
    {
        "Sid": "KinesisStreamRead",
        "Effect": "Allow",
        "Action": [
            "kinesis:GetRecords",
            "kinesis:GetShardIterator",
            "kinesis:DescribeStream",
            "kinesis:DescribeStreamSummary",
            "kinesis:ListShards",
            "kinesis:ListStreams"
        ],
        "Resource": "*"
    },
    {
      "Sid": "SecretsManagerGetSecret",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:*:*:secret:*"
      ]
    }
  ]
}
```

3. For the KMS policy, replace the *Resource* with the ARN for the KMS key created in a previous step.
4. Select **Review policy**, type in a policy name, for example, *ProtegrityLogForwarderLambdaPolicy* and **Confirm**. Record the policy name:

   LogForwarderLambdaPolicyName:_____

3.8.10 Create Log Forwarder IAM Role

Perform the following steps to create Log Forwarder execution IAM role.

➤ To create Log Forwarder IAM role:

1. From AWS IAM console, select **Roles** > **Create Role**.
2. Select **AWS Service** > **Lambda** > **Next**.
3. Select the policy created in *Create Audit Log Forwarder IAM Execution Policy*.
4. Proceed to **Name, Review and Create.**
5. Type the role name, for example, *ProtegrityForwarderRole* and click **Confirm.**
6. Record the role ARN.

   Log Forwarder IAM Execution Role Name: _____

3.8.11 Installation Artifacts

Audit Log Forwarder installation artifacts are part of the same deployment package as the one used for protect and policy agent services. Follow the steps below to ensure the right artifacts are available for log forwarder installation.

1. Verify that the Protegrity deployment package is available on your local system, if not, you can download it from the Protegrity portal.

   > **Note:** If you maintain multiple Protegrity Cloud Protectors, make sure that the deployment package downloaded for Audit Log Forwarder is the same as the one used for Protect service installation.

2. Extract the pty_log_forwarder_cf.json cloud formation file from the deployment package.
3. Check the S3 deployment bucket identified in pre-configuration steps. Make sure that all Protegrity deployment zip files are uploaded to the S3 bucket.

3.8.12 Install through CloudFormation

The following steps describe the deployment of the Audit Log Forwarder AWS cloud components.

1. Access CloudFormation and select the target AWS Region in the console.
2. Click **Create Stack** and choose **With new resources.**
3. Specify the template.
4. Select **Upload a template file.**
5. Upload the Protegrity-provided CloudFormation template called *pty_log_forwarder_cf.json* and click **Next.**
6. Specify the stack details. Enter a stack name.

   > **Note:**
   > The stack name will be appended to all the services created by the template.

7. Enter the required parameters. All the values were generated in the pre-configuration steps.

Parameter	Description	Default Value	Required
LogForwarderSubnets	Subnets where the Log Forwarder will be hosted.		
LogForwarderSecurityGroups	Security Groups, which allow communication between the Log Forwarder and ESA.		**X**
LambdaExecutionRoleArn	The ARN of Lambda role created in the prior step.		**X**
ArtifactS3Bucket	Name of S3 bucket created in the pre-configuration step.		**X**
LogDestinationEsaIp	IP or FQDN of the ESA instance or cluster.		**X**
EsaCredentialsSecretId	AWS Secrets Manager secret id where ESA Audit Store Basic Authentication credentials are stored. **Note:** *You must provide either this parameter or **PtyEsaClientCertificatesSecretId** but not both at the same time.		***X**

Parameter	Description	Default Value	Required
PtyEsaClientCertificatesSecretId	AWS Secrets Manager secret id containing client certificates used for authentication with ESA Audit Store. It is expected that the public key will be stored in a field public_key and the private key in a field named private_key. **Note:** *You must provide either this parameter or **EsaCredentialsSecretId** but not both at the same time.		***X**
EsaTlsDisableCertVerify	Disable certificate verification when connecting to ESA if set to 1. This is only for dev purposes, do not disable in production environment.	0	**X**
PtyEsaCaServerCert	ESA self-signed CA certificate used by log forwarder Lambda to ensure ESA is the trusted server. Recorded in step *Certificates on ESA* In case ESA is configured with publicly signed certificates, the PtyEsaCaServerCert configuration will be ignored.		
EsaConnectTimeout	Time in seconds to wait for the ESA response. Minimum value: 1.	5	**X**
EsaVirtualHost	ESA virtual hostname. This configuration is optional and it can be used when proxy server is present and supports TLS SNI extension.		
KinesisLogStreamRetentionPeriodHours	The number of hours for the log records to be stored in Kinesis Stream in case log destination server is not available. Minimum value: 24. See *Log Forwarder Performance* section for more details.	24	**X**
KinesisLogStreamShardCount	The number of shards that the Kinesis log stream uses. For greater provisioned throughput, increase the number of shards. Minimum value: 1. See *Log Forwarder Performance* section for more details.	10	**X**
MinLogLevel	Minimum log level for protect function. Allowed Values: off, severe, warning, info, config, all	severe	**X**

8. Click **Next** with defaults to complete CloudFormation.
9. After CloudFormation is completed, select the **Outputs** tab in the stack.
10. Record the following values

    KinesisLogStreamArn: _____

3.8.13 Add Kinesis Put Record permission to the Protect Function IAM Role

1. Login to the AWS account that hosts the Protect Lambda Function.
2. Search for Protect Lambda Function IAM Execution Role Name created in *Create Protect Lambda IAM Role*.
3. Under **Permissions policies**, select **Add Permissions** > **Create inline policy**.
4. In **Specify permissions** view, switch to **JSON**.
5. Copy the policy json from below replacing the placeholder value indicated in the following snippet as **<Audit Log Kinesis Stream ARN>** with the value recorded in the previous step.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KinesisPutRecords",
            "Effect": "Allow",
            "Action": "kinesis:PutRecords",
            "Resource": "<Audit Log Kinesis Stream ARN>"
        }
    ]
}
```

6. When you are finished, choose **Next**.
7. On the **Review and create** page, type a **Name**, then choose **Create policy**.

3.8.14 Test Log Forwarder Installation

Follow the steps to validate Log Forwarder Function installation. Successful Log Forwarder installation will aggregate logs, connect to ESA and send audit log events.

Testing in this section validates the connectivity between Log Forwarder and ESA. The sample policy included with the initial installation and test event below are not based on your ESA policy. Any logs forwarded to ESA which are not signed with a policy generated by your ESA will not be added to the audit store.

Before you begin
Install Log Forwarder and configure according to previous sections. Log Forwarder configuration **MinLogLevel** must be at least **info** level.

1. Navigate to the log forwarder lambda function.
2. Select the **Test** tab.
3. Copy the json test event into the **Event JSON** pane.

```
{
    "Records": [
        {
            "kinesis": {
                "kinesisSchemaVersion": "1.0",
                "partitionKey": "041e96d78c778677ce43f50076a8ae3e",
                "sequenceNumber":
"49620336010289430959432297775520367512250709822916263938",
                "data":
"ewogICAgICAgICAgICAiYWRkaXRpb25hbF9pbmZvIjogewogICAgICAgICAgICAgImRlc2NyaXB0aW9uIjogI
kRhdGEgdW5wcm90ZWN0IG9wZXJhdGlvbiB3YXMgc3VjY2Vzc2Z1bC4iLAogICAgICAgICAgICAgICAgInJlcXVlc3R
faWQiOiAicnMtcXVlcnktaWQ6MTIzNCIKICAgICAgICAgICAgfSwKICAgICAgICAgICAgImNsaWVudCI6IHt9LAogI
CAgICAgICAgICAiY250IjogNCwKICAgICAgICAgICAgImNvcnJlbGF0aW9uaWQiOiAicnMtcXVlcnktaWQ6MTIzNCI
```

```
sCiAgICAgICAgICAgICJsZXZlbCI6ICJTVUNDRVNTIiwKICAgICAgICAgICAgImxvZ3R5cGUiOiAiUHJvdGVjdGlvb
iIsCiAgICAgICAgICAgICJvcmlnaW4iOiB7CiAgICAgICAgICAgICAgICAiaG9zdG5hbWUiOiAiUFJPLVVTLVBGM1J
IR0Y5IiwKICAgICAgICAgICAgICAgICJ0aW1lX3V0YyI6IDE3MTE1NTkxMDAKICAgICAgICAgICAgfSwKICAgICAgI
CAgICAgInByb2Nlc3MiOiB7CiAgICAgICAgICAgICAgICAiaWQiOiAxMzA2NTYKICAgICAgICAgICAgfSwKICAgICA
gICAgICAgInByb3RlY3Rpb24iOiB7CiAgICAgICAgICAgICAgICAiYXVkaXRfY29kZSI6IDgsCiAgICAgICAgICAgI
CAgICAiZGF0YWVsZW1lbnQiOiAiYWxwaGEiLAogICAgICAgICAgICAgICAgImRhdGFzdG9yZSI6ICJTQU1QTEVfUE9
MSUNZIiwKICAgICAgICAgICAgICAgICJvcGVyYXRpb24iOiAiVW5wcm90ZWN0IiwKICAgICAgICAgICAgICAgICJwb
2xpY3lfdXNlciI6ICJQVFlERVYiCiAgICAgICAgICAgIH0sCiAgICAgICAgICAgICJwcm90ZWN0b3IiOiB7CiAgICA
gICAgICAgICAgICAiY29yZV92ZXJzaW9uIjogIjEuMi4xKzU1Lmc1OTBmZS5IRUFEIiwKICAgICAgICAgICAgICAgI
CJmYW1pbHkiOiAiY3AiLAogICAgICAgICAgICAgICAgInBjY192ZXJzaW9uIjogIjMuNC4wLjE0IiwKICAgICAgICA
gICAgICAgICJ2ZW5kb3IiOiAicmVkc2hpZnQiLAogICAgICAgICAgICAgICAgInZlcnNpb24iOiAiMC4wLjEtZGV2I
gogICAgICAgICAgICB9LAogICAgICAgICAgICAic2lnbmF0dXJlIjogewogICAgICAgICAgICAgImNoZWNrc3V
tIjogIjVDNzBCQjE2MDhENkY3NzUzMkNDQkJEQjg3RDFBREZEQTdBNEQ0MTk1RTJGNUMyMkQ5NzNEQURCNTIyQzFBR
DMiLAogICAgICAgICAgICAgICAgImtleV9pZCI6ICI4MjMyNTQ0YS1hMDVjLTRlODgtOCIKICAgICAgICAgICAgfQo
gICAgICAgIH0=",
                "approximateArrivalTimestamp": 1626878559.213
            },
            "eventSource": "aws:kinesis",
            "eventVersion": "1.0",
            "eventID":
"shardId-000000000000:496203360102894309594322977755203675122507098229162612 34",
            "eventName": "aws:kinesis:record",
            "invokeIdentityArn": "arn:aws:iam::555555555555:role/service-role/TestRole",
            "awsRegion": "us-east-1",
            "eventSourceARN": "arn:aws:kinesis:us-east-1:555555555555:stream/
CloudProtectEventStream"
        }
    ]
}
```


> **Note:** The **data** payload is the base64 encoded audit log. See *Audit Logging* for detail on audit log contents.

4. Select **Test** to execute the test event.
5. Test is successful if the Log Output of test results contains the following log:

```
[INFO] [kinesis-log-aggregation-format.cpp:77] Aggregated 1 records into 0 aggregated, 1
forwarded and 0 failed records
```

If the log is not present, please consult the *Troubleshooting* section for common errors and solutions.

3.8.15 Update Protector With Kinesis Log Stream

In this section, Kinesis log stream ARN will be provided to the Protect Function installation.

> **Note:** If the Protector has not been installed, you may provide the KinesisLogStreamArn during protector installation and skip the remainder of this section.

1. Navigate to the Protector CloudFormation stack created in the protector installation section.
2. Select **Update**.
3. Choose **Use existing template** > **Next**.
4. Set parameter **KinesisLogStreamArn** to the output value recorded in *Install through CloudFormation*.
5. Proceed with **Next** and **Submit** the changes.
6. Continue to the next section once stack status indicates **UPDATE_COMPLETE**.

3.8.16 Update Policy Agent With Log Forwarder Function Target

Log Forwarder Lambda function requires a policy layer which is in sync with the Protegrity Protector. This section will describe the steps to update the policy agent to include updating Log Forwarder Lambda function.

> **Note:** If the policy agent has not been installed, follow the steps in *Policy Agent Installation*. Set **AWS_PROTECT_FN_NAME** to include both protector and log forwarder lambda functions.

1. Navigate to the Policy Agent Function created in *Policy Agent Installation*
2. Select **Configuration** > **Environment variables** > **Edit**
3. Edit the value for environment variable **AWS_PROTECT_FN_NAME** to include the log forwarder function name/arn in the comma separated list of Lambda functions.
4. Save the changes and continue when update completes
5. Navigate to **Test** tab
6. Add an event **{}** and select **Test** to run the Policy Agent function
7. Verify Log forwarder function was updated to use the policy layer by inspecting the log output. Logs should include the following:

```
[INFO] 2024-07-09 18:58:04,793.793Z 622d374b-1f73-4123-9a38-abc61973adef
iap_agent.policy_deployer:Updating lambda [Protegrity_LogForwarder_<stack ID>] to use
layer version [arn:aws:lambda:<aws region>:<aws account
number>:layer:Protegrity_Layer_<layer name>:<layer version>]
```

3.8.17 Test Full Log Forwarder Installation

Before you begin
Install and configure Protegrity Agent, Protector, and Log Forwarder components.

1. Send a protect operation to the protector using a data element or user which will result in audit log generation
2. Navigate to the CloudWatch log group for the Protect function
3. Select the log stream for the test operation and scroll to the latest logs
4. Expect to see a log similar to the below:

```
[2024-07-09T19:28:23.158] [INFO] [kinesis-external-sink.cpp:51] Sending 2 logs to
Kinesis ...
[2024-07-09T19:28:23.218] [INFO] [aws-utils.cpp:206] Kinesis send time: 0.060s
```

5. Navigate to the CloudWatch log group for the Log Forwarder function
6. Expect to see a new log stream - it may take several minutes for the stream to start
7. Select the new stream and scroll to the most recent logs in the stream
8. Expect to see a log similar to the below:

```
[2024-07-09T19:32:31.648] [INFO] [kinesis-log-aggregation-format.cpp:77] Aggregated 1
records into 0 aggregated, 1 forwarded and 0 failed records
```

3.8.18 Troubleshooting

Error	Action
Log forwarder log contains severe level secrets permissions error: `[SEVERE] User: <arn> is not authorized to perform: secretsmanager:GetSecretValue on resource: <secret name> because no identity-based policy allows the secretsmanager:GetSecretValue action`	1. Verify the permission policy/role attached to the log forwarder function has secretsmanager:GetSecretValue permission for the insights esa user credentials secret. 2. Consult sections *Configure ESA Audit Store Credentials* and *Create Audit Log Forwarder IAM Execution Policy*
When testing log forwarder as described in *Test Log Forwarder Installation*, response contains policy decryption error: `{ "error_msg": "Failed to decrypt the policy. Please verify that the function has access to the key service and the key.", "success": false }`	1. Verify the permission policy/role attached to the log forwarder function has kms:Decrypt permission for KMS key used to encrypt the Protegrity security policy. 2. Consult section *Create Audit Log Forwarder IAM Execution Policy*
Cloudformation stack creation fails with error: `The provided execution role does not have permissions to call [CreateNetworkInterface|DescribeNetworkInterfaces|DeleteNetworkInterface] on EC2 (Service: Lambda, Status Code: 400, Request ID: <request id>)" (RequestToken: <request token>, HandlerErrorCode: InvalidRequest)`	1. Verify the permission policy/role attached to the log forwarder function has ec2:CreateNetworkInterface, ec2:DescribeNetworkInterfaces, ec2:DeleteNetworkInterface permissions 2. Consult section *Create Audit Log Forwarder IAM Execution Policy*
Severe level kinesis permissions log message in protector function: `[SEVERE] Kinesis stream client returned 400 error with error message: User: <function arn> is not authorized to perform: kinesis:PutRecords on resource: <kinesis stream arn> because no identity-based policy allows the kinesis:PutRecords action`	1. Verify the permission policy/role attached to the protector function has kinesis:PutRecords permission 2. Consult section *Add Kinesis Put Record permission to the Protect Function IAM Role*
TLS errors reported in log forwarder function logs: `[error] [tls] <error message>`	1. If ESA is using self-signed certificate, verify the correct ESA certificate has been given in the format described in *Certificates on ESA*

Chapter 4

REST API Authorization

4.1 Policy Users

Protegrity Policy roles defines the unique data access privileges for every member. The Protegrity Lambda protects the data with the username sent in either the JWT-formatted authorization header or the request body.

The lambda behavior can be set in the Lambda environment variables as described in *Protect Lambda Configuration*

Authorization/allow_assume_user	0	1
Empty	User from the request body. / (Throw an error).	User from the request body.
JWT	User from JWT payload	User from request body. If not found user from JWT payload.

4.2 JWT Verification

To ensure the integrity of the user, the lambda protect can verify the JWT.

1. From your AWS console, navigate to lambda and select the following Lambda: **Protegrity_Protect_RESTAPI_<STACK_NAME>**
2. Scroll down to the **Environment variables** section, select **Edit** to replace the entries.

Parameter	Value	Notes
authorization	JWT	
jwt_verify	1	

jwt_secret_base64	Secret in base64 encoding. For example, the value of the public key is as follows.
	`-----BEGIN PUBLIC KEY-----`
	`MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC4fkg/`
	`JYyN3Skr6RYLiAd/Yhl0`
	`2TE3/HzHSNPnCaRdUakGp9og7oXBMcoadFDjnoSq1sz`
	`+gUHnpoO7s2fwkD5Q4OnC`
	`BGD3oKP2A4PlOOWD2B2cVmMqX/vf1nAA/`
	`343496jsbfgkh1Q7LTzR0IXfdii0o1U`
	`CbvrVCuaBoyiv4TxWQIDAQAB`
	`-----END PUBLIC KEY-----`
		The secret must be in **base64**. We recommend using RSA public certificates, it is not recommended to keep Hash (symmetric) secrets in the clear.

Parameter	Value	Notes

	This public key will be stored as follows.
	
	`LS0tLS1CRUdJTiBQVUJMSUMgS0VZLS0tL`
	`S0KTUlHZk1BMEdDU3FHU0liM0RRRUJBUV`
	`VBQTRHTkFEQ0JpUUtCZ1FDNGZrZy9KWXl`
	`OM1NrcjZSWUxpQWQvWWhsMAoyVEUzL0h6`
	`SFNOUG5DYVJkVWFrR3A5b2c3b1hCTWNvY`
	`WRGRGpub1NxMXN6K2dVSG5wb083czJmd2`
	`tENVE0T25DCkJHRDNvS1AyQTRQbE9PV0Q`
	`yQjJjVm1NcVgvdmYxbkFBLzM0MzQ5Nmpz`
	`YmZna2gxUTdMVHpSMElYZmRpaTBvMVUKQ`
	`2J2clZDdWFCb3lpdjRUeFdRSURBUUFCCi`
	`0tLS0tRU5EIFBVQkxJQyBLRVktLS0tLQ==`	

Chapter 5

REST API

The following sections describe the key concepts in understanding the REST API.

Base URL:

```
https://{ApiGatewayId}.execute-api.{Region}.amazonaws.com/pty/v1
```

5.1 Export a REST API from API Gateway

Once Protegrity Serverless REST API is installed, you can export the OpenAPI documentation file from API Gateway Export API, located in the AWS API Gateway Control Service.

https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-export-api.html

For testing the REST API, we recommend using tools, such as, Postman.

5.2 Payload v1

The following encoding formats are supported in the REST API.

For every encoding, the resultant protected data is returned in the same encoding. For example, if request is hex-encoded, the response is also hex-encoded.

For more information about the encoding formats, refer to the *Protection Method References Guide.*

Encoding	Supported by data elements	Notes
utf8	All except binary data elements.	Default encoding if encoding is not specified.
hex	All	Default encoding for binary data elements.
base64	All	• *Uses RFC 4648 Section 4 Alphabet.* • Accepts input of any length without line separators. • Optional padding.

Encoding	Supported by data elements	Notes
		• Does not accept any non-base64 characters.
base64_mime	All	• *Uses RFC 2045 Alphabet.* • Accepts input on multiple lines. • Lines can be separated by any combination of CR and LF characters. • Lines must not be longer than 76 characters long. • Response lines are separated by CRLF characters. • Optional padding. • Does not accept any non-base64 characters.
base64_pem	All	• *Uses RFC 1421 Alphabet.* • Accepts input on multiple lines. • Lines can be separated by any combination of CR and LF characters. • Lines must not be longer than 64 characters long. • Response lines are separated by CRLF characters. • Optional padding. • Does not accept any non-base64 characters.
base64_url	All	• *Uses RFC 4648 Section 5 Alphabet.* • Accepts input of any length without line separators. • Optional padding. • Does not accept any non-base64 characters.

The following sections provide the payload schema and its examples.

5.2.1 Request

AWS Lambda service limits maximum size of payload to 6 MB. Client applications of Protegrity Cloud API must ensure their payload size is within this limit. This applies to all types of requests described below.

Performs a policy operation such as protect, unprotect, or reprotect.

URI

/v1/protect or ***/v1/unprotect*** or ***/v1/reprotect***

Method

POST

Parameters

data: Input data to the policy operation.

data_element: Data element to use for the policy operation.

encoding: Optional, encoding of the data. One of: base64, base64_mime, base64_pem, base64_url, hex, or utf8. Defaults to hex for binary data elements, otherwise defaults to utf8.

external_iv: Optional, external initialization vector.

old_data_element: (reprotect) Data element for unprotecting the input.

old_external_iv: (reprotect) Optional, external initialization vector for the input.

query_id: Optional, identifier for the request.

user: User performing the operation.

Result

Returns the output of the policy operation.

Example 1 - without external IV

```
{
  "encoding": "utf8",
  "query_id": "1",
  "user": "user1",
  "data_element": "Alphanum",
  "data":[<data1>,<data2>]
}
```

Example 2 - with external IV

```
{
  "encoding": "utf8",
  "query_id": "1",
  "user": "user1",
  "data_element": "Alphanum",
  "external_iv": "abc123",
  "data":[<data1>,<data2>]
}
```

Example 3 - reprotect

```
{
  "encoding": "utf8",
  "query_id": "1",
  "user": "user1",
  "data_element": "deName",
  "old_data_element": "Alphanum",
  "data":[<data1>,<data2>]
}
```

Performs policy operations using different data elements for each data set.

URI

/v1/protect or ***/v1/unprotect*** or ***/v1/reprotect***

Method

POST

Parameters

encoding: Optional, encoding of the data. One of: base64, base64_mime, base64_pem, base64_url, hex, or utf8. Defaults to hex for binary data elements, otherwise defaults to utf8.

query_id: Optional, prefix for the request identifier.

user: User performing the operation.

arguments[].data: Input data to the policy operation.

arguments[].data_element: Data element to use for the policy operation.

arguments[].external_iv: Optional, external initialization vector.

arguments[].id: Optional, request identifier.

arguments[].old_data_element: (reprotect) Data element for unprotecting the input.

arguments[].old_external_iv: (reprotect) Optional, external initialization vector for the input.

> **Note:** When multiple data elements are sent in the payload, the Cloud API generates an audit log per argument. The Cloud API appends the argument id, if it exists, to the query_id. For example: *query_id.id:{id}*. If an argument id is not provided then the index in the argument array is appended to the query_id. For example: *query_id.index:{index}*.

Example 4 - multiple data elements support payload

```
{
  "encoding": "utf8",
```

```
  "query_id": "query1234",
  "user": <policy_user>,
  "arguments": [
    {
      "id": "1",
      "external_iv": "<external iv>",
      "data_element": "<data element name>",
      "data":["<data1>","<data2>"]
    },
    {
      "data_element": <data element name>,
      "data":["<data1>","<data2>"]
    }
  ]
}
```

External IV

The External IV feature provides an additional level of security by allowing different tokenized results across protectors for the same input data and token element, depending on the External IV set on each protector. External IVs must adhere to certain guardrails and not all data elements support External IV. To read more about the Tokenization model with External IV, refer to the External Initialization Vector (IV) chapter in the *Protection Methods Reference Guide*.

5.2.2 Response

```
responsePayloadV1:
type: object
  properties:
    success:
      type: bool
    error_msg:
      type: string
      description: When success is false, error_msg is included
    results:
      type: array
      items:
        type: string
```

Example success:

```
{
  "encoding": "utf8",
  "results":["str1","str2"],
  "success": true
}
```

If the request was successful, the success flag will always be true.

Example failure:

```
{
  "error_msg": "token expired",
  "success": false
}
```

If the request fails, the success flag will always be false.

Multi Data Elements Support Payload

```
responsePayloadV2:
type: object
  properties:
    success:
      type: boolean
    results:
      type: array
```

```
        items:
          type: object
          properties:
            success:
              type: bool
            error_msg:
              type: string
              description: When success is false, error_msg is included
            id:
              type: string
              description: When id is sent in the request payload, id is included in the
response
            results:
              type: array
              items:
                type: string
```

Example success:

```
{
  "encoding": "utf8",
  "results":[
    {
      "encoding": "utf8",
      "results":["str1","str2"],
      "success": true
    },
    {
      "encoding": "utf8",
      "results":["str1","str2"],
      "success": true
    }
  ],
  "success": true
}
```

If the all the subrequests were successful, the success flag will be true.

Example failure:

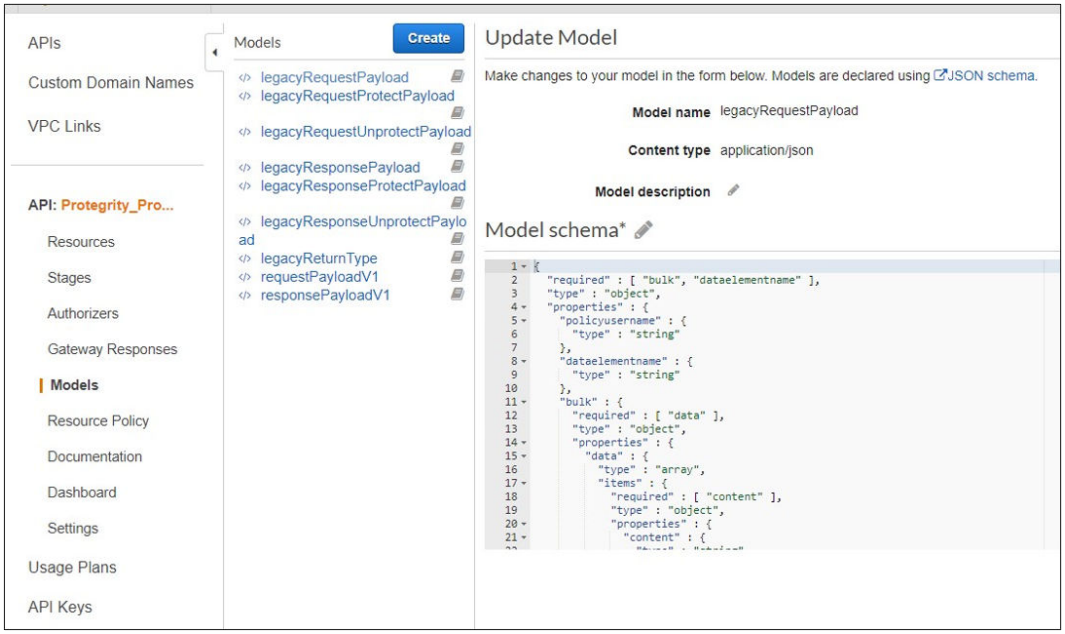
```
{
  "results": [
    {
      "error_msg":
      "Protect failed. Data element not found. Refer to audit log for details",
      "success": false
    },
    {
      "encoding": "utf8",
      "results":["str1","str2"],
      "success": true
    }
  ],
  "success": false
}
```

It is possible to have a mix of successful and unsuccessful results. In this case, the global success flag will be false.

5.3 Legacy

Protegrity has multiple products with REST API capabilities, such as Protection Server (out of support), DSG, and the latest product - IAP REST. Each one has its use case. To help you move to cloud-native implementation, Cloud Product REST API supports legacy payload.

Legacy documentation can be downloaded from AWS console, API, Models, prefix Legacy.



5.3.1 Request

Performs a policy operation such as protect, unprotect, or reprotect.

Method

POST

Parameters

dataelementname: (protect/unprotect) Data element to use for the policy operation.

externaliv: (protect/unprotect) Optional, external initialization vector.

newdataelementname: (reprotect) Data element to use for the output.

newexternaliv: (reprotect) Optional, external initialization vector for the output.

olddataelementname: (reprotect) Data element to use for the input.

oldexternaliv: (reprotect) Optional, external initialization vector for the input.

policyusername: User performing the operation.

bulk.id: Optional, identifier for the request.

bulk.data[].content: Input data to the policy operation.

Result

Returns the output of the policy operation.

Example 1 - protect without external IV

```
{
  "protect": {
    "policyusername": "user1",
    "dataelementname": "Alphanum",
    "bulk": {
      "id": "1",
      "data": [
        {
          "content": <Data encoded in base64>
        }
      ]
    }
  }
}
```

Example 2 - protect with external IV

```
{
  "protect": {
    "policyusername": "user1",
    "dataelementname": "Alphanum",
    "externaliv": "abc123",
    "bulk": {
      "id": "1",
      "data": [
        {
          "content": <Data encoded in base64>
        }
      ]
    }
  }
}
```

Example 3 - unprotect

```
{
  "unprotect": {
    "policyusername": "user1",
    "dataelementname": "Alphanum",
    "bulk": {
      "id": "1",
      "data": [
        {
          "content": <Data encoded in base64>
        }
      ]
    }
  }
}
```

Example 4 - reprotect

```
{
  "reprotect": {
    "policyusername": "user1",
    "newdataelementname": "deName",
    "olddataelementname": "Alphanum",
    "bulk": {
      "id": "1",
      "data": [
        {
          "content": <Data encoded in base64>
        }
      ]
    }
  }
}
```

5.3.2 Response

Example:

```
{"protect":{"bulk":{"returntype":"success","data":[{"returntype":"success","message":"Data
 protection was successful.","content":"RGZBUFR4ODAzejFwNjQ5TWg0TEFpcFNqbA=="},
{"returntype":"success",
"message":"Data protection was successful.","content":"aHNnVVB5QWFDYw=="}]}}}
```

5.4 HTTP Status Codes

The following table explains the different HTTP Status Codes with their corresponding response.

Status Codes	Response	Description
200	`{"results": ["<string>","<string>"], "success": true, "encoding":[hex|utf8]}`	Success protected data is in results, and success attribute is true
400	`{ "error_msg": "<string>", "success": false }`	There was an issue in the request, success is false, check error_msg attribute. For more information check AWS Lambda's CloudWatch logs. For example, the following error appears. `Authorization header was not found.` 1. Required attribute is missing /user, when env authorization is None or empty. 2. User is missing, check JWT payload for user attribute 3. Malformed authorization header 4. Basic Not supported in header: authorization

403	`{ "error_msg": "token expired", "success": false }`
	
	`{ "error_msg": "invalid signature", "success": false }`
	`{ "error_msg": " unknown algorithm error", "success": false }`
		1. JWT is invalid because secret is wrong or expired.
		2. Get A new JWT or check the base64 secret in AWS Lambda environment variable.
		3. Secret doesn't match the algorithm type in the JWT header. Secret is symmetric but Algorithm is asymmetric.

500	`{ "error_msg": "Invalid json array in jwt_user_claim", "success": false }`
	
	`{ "error_msg": "jwt secret is missing ", "success": false }`
		1. Configuration error in AWS Lambda.
		Authorization is to JWT and jwt _user_claim is not a valid json array. Example valid input: ["username", "firstname"].
		2. JWT is set to verify and jwt_secret_base64 is missing.

5.5 TLS Mutual Authentication

By default, AWS API Gateway supports HTTPS endpoint and doesn't allow HTTP protocol.

For an additional layer of security, you can configure AWS API Gateway TLS Mutual Authentication.

AWS API Gateway will ensure that the clients with a valid certificate only can call the REST API. Protegrity recommends to setup TLS Mutual Authentication in AWS API Gateway. For more information on how to set AWS API Gateway Mutual TLS go to the following link.

https://docs.aws.amazon.com/apigateway/latest/developerguide/rest-api-mutual-tls.html

Chapter 6

Performance

6.1 Performance Considerations

The following factors may cause variation in real performance versus benchmarks:

1. Cold startup: The Lambda spends additional time on the initial invocation to decrypt and load the policy into memory. This time can vary between 400 ms and 1200 ms depending on the policy size. Once the Lambda is initialized, subsequent “warm executions” should process quickly.
2. Noisy neighbors: There are many multi-tenant components in the Cloud. The same request may differ by 50% between runs regardless of Protegrity. A single execution may not be the best predictor of average performance.
3. Size of policy: The size of the policy impacts cold start performance. Larger policies take more time to initialize.
4. Lambda memory: AWS provides more virtual cores based on the memory configuration. The initial configuration of 1728MB provides a good tradeoff between performance and cost with the benchmarked policy. Memory can be increased to optimize for your individual case.
5. Number of protect, unprotect and reprotect security operations.
6. Lambda Concurrent and Burst quotas: AWS limits the number of concurrent executions and how quickly lambda can scale to meet demand. This is discussed in an upcoming section of the document.
7. Size of data element. Operations on larger text may take more time.
8. API Gateway Authentication. API Gateway support different authorizers and authentication method, which may differ.

6.2 Sample Benchmarks

6.2.1 Lambda Tuning

AWS maintains quotas for Lambda concurrent execution. Two of these quotas impact concurrency and compete with other Lambdas in the same account and region.

Service quota	AWS default quota value	Adjustable
Concurrent executions	1,000 / Second	Yes
Burst concurrency	3,000	No

The *concurrent executions* quota cap is the maximum number of Lambda instances that can serve requests for an account and region. The default AWS quota may be inadequate based on peak concurrency based on the table in the previous section. This quota can be increased with an AWS support ticket.

The *Burst concurrency* quota limits the rate at which Lambda will scale to accommodate demand. This quota is also per account and region. The burst quota cannot be adjusted. AWS will quickly scale until the burst limit is reached. After the burst limit is reached, functions will scale at a reduced rate per minute (e.g. 500). If no Lambda instances can serve a request, the request will fail with a 429 Too Many Requests response.

The burst limit is a fixed value and varies significantly by AWS region. The highest burst (3,000) is currently available in the following regions: US West (Oregon), US East (N.Virginia), and Europe (Ireland). Other regions can burst between 500 and 1,000. It is recommended to select an AWS region with the highest burst limits.

6.2.2 API Gateway Tuning

AWS maintains a Throttle quota for the API Gateway. By default, API Gateway limits concurrent requests to 10,000 requests per second and throttles subsequent traffic with a 429 Too Many Requests error response. This quota applies across all APIs in an account and region.

The API Gateway default quota may need to be increased based on the Concurrency table. Keep in mind that hitting quota limits effectively throttles any other API services in the region.

The API Gateway also limits burst. Burst is the maximum number of concurrent requests that API Gateway can fulfill at any instant without returning 429 Too Many Requests error responses. This limit can be increased by AWS, but is not adjustable.

Enable CloudWatch metrics within the API Gateway to monitor max concurrency and investigate throttling errors. See the *Concurrency Troubleshooting* section on interpreting CloudWatch metrics.

Quotas adjustments are applied for region and account. Throttling is also enabled by default in the API Gateway stage used by the Protegrity Lambda function. The stage configuration throttling must be adjusted if the quota is modified. Stage throttling is shown in the following image.

For example, the REST Client makes over 20,000 requests/sec to execute the given query. Using API Gateway's default settings, the requests exceeding 10,000 requests/sec will be throttled. Therefore, this query may fail intermittently due to a high number of throttling errors.

6.2.3 Concurrency Troubleshooting

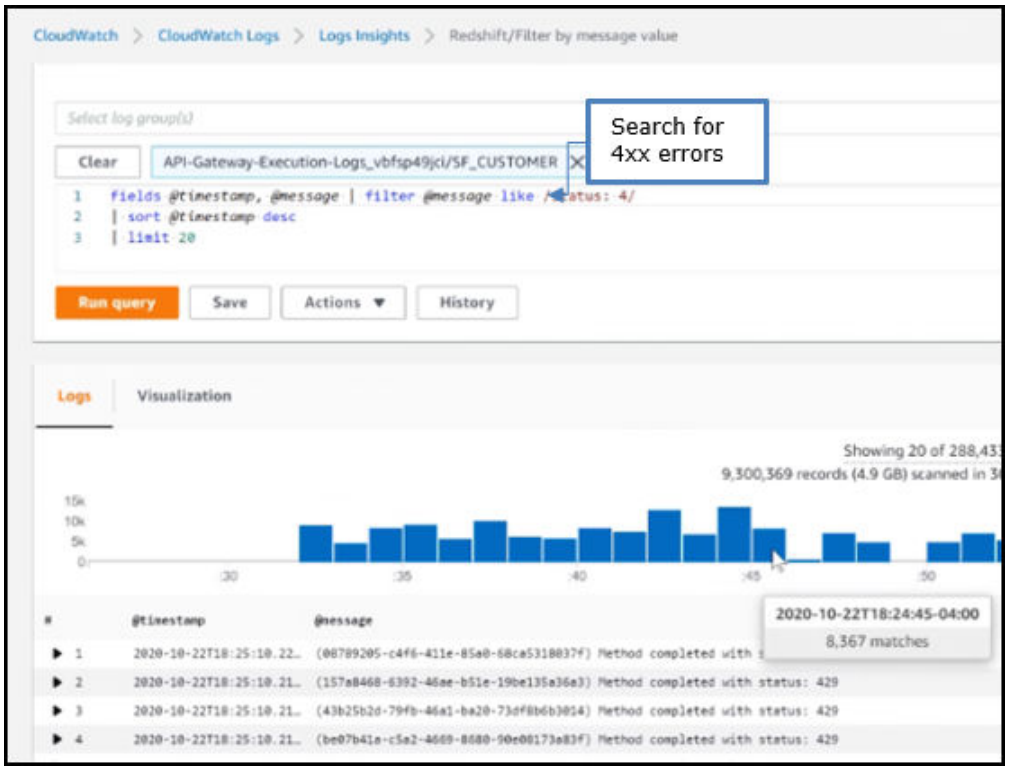
Hitting up against quota limits may indicate that quota adjustments are required. Exceeding quota limits may cause a client query to fail or reduce performance. In the worst case, significant throttling can impact the performance of all your API Gateway or Lambda services in the region.

CloudWatch Metrics can be manually enabled on the API Gateway to reveal if quotas are being reached. Metrics aggregate errors into two buckets that are 4xx and 5xx. CloudWatch logs can be used to access the actual error code. The following table describes how to interpret the CloudWatch metrics.

Error Type	Possible issue	Remedy
4xx errors	API Gateway burst or throttle quota exceeded	Request an increase to the API Gateway throttle quota.
5xx errors	Lambda concurrent requests or burst quota exceeded. Verify 4xx errors in Lambda Metrics.	Request an increase the Lambda concurrent request quota

Note:
The API Gateway Lambda proxy maps 429 errors from the Lambda service to 500 errors.

The following screenshot shows an example of searching CloudWatch Logs using Log Insights:



6.2.4 Cold-Start Performance

Cold-start vs warm execution refers to the state of the Lambda when a request is received. A cold-start undergoes additional initialization, such as, loading the security policy. Warm execution applies to all subsequent requests served by the Lambda.

The following table shows an example how these states impact latency and performance:

Execution state	Avg. Execution Duration	Avg. Total (w/ network latency)
Cold execution	438 ms	522 ms
Warm execution	< 2ms	84 ms

Note:
Cold execution time will vary based on the physical size of the security policy. A large security policy will result in longer cold startup times.

6.3 Log Forwarder Performance

Log forwarder architecture is optimized to minimize the amount of connections and reduce the overall network bandwidth required to send audit logs to ESA. This is achieved with batching and aggregation taking place on two levels. The first level is in protect function instances, where audit logs from consecutive requests to an instance are batched and aggregated. The second level of batching takes place in Amazon Kinesis Stream where log records from different protect function instances are additionally batched and sent to log forwarder function where they are aggregated. This section shows how to configure the deployment to accommodate different patterns of anticipated audit log stream. It also shows how to monitor deployment resources to detect problems before audit records are lost.

Protector Cloud Formation Parameters

- **AuditLogFlushInterval:** Determines the minimum amount of time required for the audit log to be sent to Amazon Kinesis. Changing flush interval may affect the level of aggregation, which in turn may result in different number of connections and different data rates to Amazon Kinesis. Default value is 30 seconds.
  Increasing the flush interval may result in higher aggregation of audit logs, in fewer connections to Amazon Kinesis, in higher latency of audit logs arriving to ESA and in higher data throughput.
  Lowering the flush interval may result in lower aggregation of audit logs, in more connections to Amazon Kinesis, in lower latency of audit logs arriving to ESA and in lower data throughput.
  It is not recommended to reduce the flush interval from default value in production environment as it may overload the Amazon Kinesis service. However, it may be beneficial to reduce flush interval during testing to make audit records appear on ESA faster.

Log Forwarder Cloud Formation Parameters

- **Amazon KinesisLogStreamShardCount:** The number of shards represents the level of parallel streams in the Amazon Kinesis and it is proportional to the throughput capacity of the stream. If the number of shards is too low and the volume of audit logs is too high, Amazon Kinesis service may be overloaded and some audit records sent from protect function may be lost.
  Default value is 10, however you are advised to test with a production-like load to determine whether this is sufficient or not.
- **Amazon KinesisLogStreamRetentionPeriodHours:** The time for the audit records to be retained in Amazon Kinesis log stream in cases where log forwarder function is unable to read records from the Kinesis stream or send records to ESA, for example due to a connectivity outage. Amazon Kinesis will retain failed audit records and retry periodically until connectivity with ESA is restored or retention period expires.
  Default value is 24 hours, however you are advised to review this value to align it with your Recovery Time Objective and Recovery Point Objective SLAs.

Monitoring Log Forwarder Resources

- **Amazon Kinesis Stream Metrics:** Any positive value in Amazon Kinesis PutRecords throttled records metric indicates that audit logs rate from protect function is too high. The recommended action is to increase the Amazon KinesisLogStreamShardCount or optionally increase the AuditLogFlushInterval.

- **Log Forwarder Function CloudWatch Logs**: If log forwarder function is unable to send logs to ESA, it will log the following message:

  ```
  [SEVERE] Dropped records: x.
  ```

  > **Note:** When the error message above occurs, the dropped audit records will be preserved in the Amazon Kinesis data stream and retried again according to Amazon Kinesis retry schedule. Records will be retried until Amazon KinesisLogStreamRetentionPeriodHours expires.

- **Protect Function CloudWatch Logs**: If protect function is unable to send logs to Amazon Kinesis, it will log the following message:

  ```
  [SEVERE] Amazon Kinesis error, retrying in x ms (retry: y/z) ..."
  ```

  Any dropped audit log records will be reported with the following log message:

  ```
  [SEVERE] Failed to send x/y audit logs to Amazon Kinesis.
  ```

Chapter 7

Audit Logging

Audit records and application logs stream to Amazon CloudWatch Logs or optionally be sent to ESA. Cloud Protect uses a JSON format for audit records that is described in the following sections.

You can analyze and alert on audit records using Protegrity ESA or Amazon CloudWatch. Third-party solutions may be used if they are supported by Amazon Cloudwatch or AWS Lambda logging extensions. For more information about forwarding your audit records to ESA, contact Protegrity. For more information about Amazon CloudWatch, refer to the *Amazon CloudWatch User Guide*.

For more information about audit records, refer to the *Protegrity Analytics Guide*.

7.1 Audit record fields

The audit record format has been altered in version 3.1 of the protector to provide more information.

Field	Description
additional_info.cluster	(Optional) Redshift cluster ARN
additional_info.description	A human-readable message describing the operation
additional_info.query_id	(Optional) Identifies the query that triggered the operation
additional_info.request_id	(Optional) AWS Lambda request identifier
cnt	Number of operations, may be aggregated
correlationid	(Deprecated) Use additional_info instead
level	Log severity, one of: SUCCESS, WARNING, ERROR, EXCEPTION
logtype	Always "Protection"
origin.hostname	Hostname of the system that generated the log entry
origin.time_utc	UTC timestamp when the log entry was generated
protection.audit_code	Audit code of the protect operation; see the log return codes table in the *Protegrity Troubleshooting Guide*
protection.dataelement	Data element used for the policy operation
protection.datastore	Name of the data store corresponding to the deployed policy
protection.mask_setting	(Optional) Mask setting from policy management
protection.operation	Operation type, one of: Protect, Unprotect, Reprotect

Field	Description
protection.policy_user	User that performed the operation
protector.core_version	Internal core component version
protector.family	Always "cp" for Cloud Protect
protector.lambda_version	Protector Lambda application version.
protector.pcc_version	Internal pcc component version
protector.vendor	Identifies the cloud vendor and the database vendor
protector.version	Protector version number
signature.checksum	Hash value of the signature key ID used to sign the log message when the log is generated
signature.key_id	Key used to sign the log message when the log is generated

The following are sample audit messages:

Protect Success:

```
{
      "additional_info": {
        "description": "Data protect operation was successful.",
        "query_id": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
        "request_id": "8476a536-e9f4-11e8-9739-2dfe598c3fcd"
      },
      "cnt": 4000,
      "correlationid": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
      "logtype": "Protection",
      "level": "SUCESS",
      "origin": {
        "hostname": "localhost",
        "time_utc": 1635363966
      },
      "protection": {
        "dataelement": "deAddress",
        "operation": "Protect",
        "audit_code": 6,
        "datastore": "SAMPLE_POLICY",
        "policy_user": "test_user"
      },
      "client": {},
      "protector": {
        "family": "cp",
        "lambda_version": "3.2.10",
        "version": "3.2.0",
        "vendor": "aws.snowflake",
        "pcc_version": "3.4.0.14",
        "core_version": "1.2.1+55.g590fe.HEAD"
      },
      "signature": {
        "key_id": "95f5a194-b0a4-4351-a",
        "checksum": "B324AF7C56944D91C47847A77C0367C594C0B948E7E75654B889571BD4F60A71"
      }
    }
```

User permission denied:

```
{
      "additional_info": {
        "description": "The user does not have the appropriate permissions to perform the
requested operation."
      },
      "cnt": 4000,
      "correlationid": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
```

```
      "logtype": "Protection",
      "level": "ERROR",
      "origin": {
        "hostname": "localhost",
        "time_utc": 1635363966
      },
      "protection": {
        "dataelement": "deAddress",
        "operation": "Protect",
        "audit_code": 3,
        "policy_user": "test_user"
      },
      "process": {
        "id": "1",
        "thread_id": "849348352"
      },
      "client": {},
      "protector": {
        "family": "IAP Lambda",
        "lambda_version": "3.2.10",
        "version": "3.2.0",
        "vendor": "Cloud Protect",
        "pcc_version": "3.3.0.5",
        "core_version": "1.1.0"
      },
      "signature": {
        "key_id": "95f5a194-b0a4-4351-a",
        "checksum": "A216797C56944D91C47847A77C0367C594C0B948E7E75654B889571BD4F60A71"
      }
    }
```


Data element not found:


```
{
      "additional_info": {
        "description": "The data element could not be found in the policy in shared memory."
      },
      "cnt": 4000,
      "correlationid": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
      "logtype": "Protection",
      "level": "ERROR",
      "origin": {
        "hostname": "localhost",
        "time_utc": 1635363966
      },
      "protection": {
        "dataelement": "deAddress",
        "operation": "Protect",
        "audit_code": 2,
        "policy_user": "test_user"
      },
      "process": {
        "id": "1",
        "thread_id": "849348352"
      },
      "client": {},
      "protector": {
        "family": "IAP Lambda",
        "lambda_version": "3.2.10",
        "version": "3.2.0",
        "vendor": "Cloud Protect",
        "pcc_version": "3.3.0.5",
        "core_version": "1.1.0"
      },
      "signature": {
        "key_id": "95f5a194-b0a4-4351-a",
        "checksum": "AF09217C56944D91C47847A77C0367C594C0B948E7E75654B889571BD4F60A71"
      }
    }
```

7.2 Example Audit Records

The following are sample audit messages:

Protect Success:


```
{
      "additional_info": {
        "description": "Data protect operation was successful.",
        "query_id": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
        "request_id": "8476a536-e9f4-11e8-9739-2dfe598c3fcd"
      },
      "cnt": 4000,
      "correlationid": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
      "logtype": "Protection",
      "level": "SUCESS",
      "origin": {
        "hostname": "localhost",
        "time_utc": 1635363966
      },
      "protection": {
        "dataelement": "deAddress",
        "operation": "Protect",
        "audit_code": 6,
        "datastore": "SAMPLE_POLICY",
        "policy_user": "test_user"
      },
      "client": {},
      "protector": {
        "family": "cp",
        "version": "3.1.0",
        "vendor": "aws.snowflake",
        "pcc_version": "3.4.0.14",
        "core_version": "1.2.1+55.g590fe.HEAD"
      },
      "signature": {
        "key_id": "95f5a194-b0a4-4351-a",
        "checksum": "B324AF7C56944D91C47847A77C0367C594C0B948E7E75654B889571BD4F60A71"
      }
    }
```


Reprotect Success:


```
{
      "additional_info": {
        "description": "Data reprotect operation was successful.",
        "query_id": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
        "request_id": "8476a536-e9f4-11e8-9739-2dfe598c3fcd"
      },
      "cnt": 4000,
      "correlationid": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
      "logtype": "Protection",
      "level": "SUCCESS",
      "origin": {
        "hostname": "localhost",
        "time_utc": 1635363966
      },
      "protection": {
        "old_dataelement": "deAddress1",
        "dataelement": "deAddress2",
        "operation": "Reprotect",
        "audit_code": 50,
        "datastore": "SAMPLE_POLICY",
        "policy_user": "test_user"
      },
      "client": {},
      "protector": {
        "family": "cp",
        "version": "3.1.0",
```

```
        "vendor": "aws.snowflake",
        "pcc_version": "3.4.0.14",
        "core_version": "1.2.1+55.g590fe.HEAD"
      },
      "signature": {
        "key_id": "95f5a194-b0a4-4351-a",
        "checksum": "B324AF7C56944D91C47847A77C0367C594C0B948E7E75654B889571BD4F60A71"
      }
    }
```


User permission denied:


```
{
      "additional_info": {
        "description": "The user does not have the appropriate permissions to perform the
requested operation.",
        "query_id": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
        "request_id": "8476a536-e9f4-11e8-9739-2dfe598c3fcd"
      },
      "cnt": 4000,
      "correlationid": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
      "logtype": "Protection",
      "level": "ERROR",
      "origin": {
        "hostname": "localhost",
        "time_utc": 1635363966
      },
      "protection": {
        "dataelement": "deAddress",
        "operation": "Protect",
        "audit_code": 3,
        "datastore": "SAMPLE_POLICY",
        "policy_user": "test_user"
      },
      "client": {},
      "protector": {
        "family": "cp",
        "version": "3.1.0",
        "vendor": "aws.snowflake",
        "pcc_version": "3.4.0.14",
        "core_version": "1.2.1+55.g590fe.HEAD"
      },
      "signature": {
        "key_id": "95f5a194-b0a4-4351-a",
        "checksum": "A216797C56944D91C47847A77C0367C594C0B948E7E75654B889571BD4F60A71"
      }
    }
```


Data element not found:

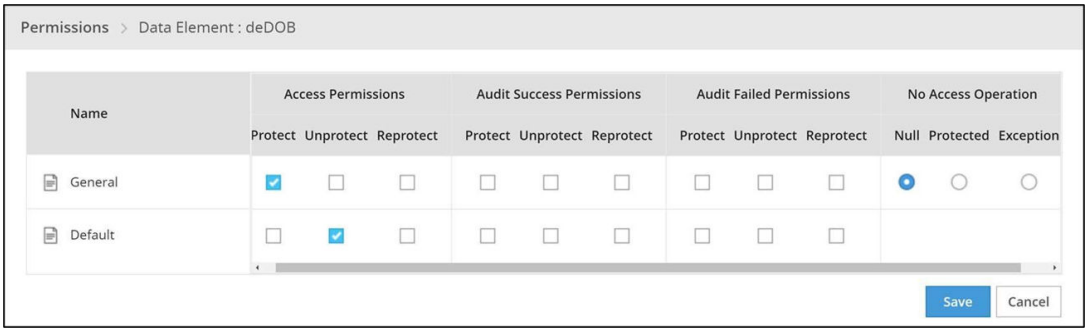

```
{
      "additional_info": {
        "description": "The data element could not be found in the policy in shared memory.",
        "query_id": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
        "request_id": "8476a536-e9f4-11e8-9739-2dfe598c3fcd"
      },
      "cnt": 4000,
      "correlationid": "sf-query-id:01978dbc-0582-d7e4-0000-002a3603a20d",
      "logtype": "Protection",
      "level": "ERROR",
      "origin": {
        "hostname": "localhost",
        "time_utc": 1635363966
      },
      "protection": {
        "dataelement": "deAddress",
        "operation": "Protect",
        "audit_code": 2,
        "datastore": "SAMPLE_POLICY",
        "policy_user": "test_user"
```

```
      },
      "client": {},
      "protector": {
        "family": "cp",
        "version": "3.1.0",
        "vendor": "aws.snowflake",
        "pcc_version": "3.4.0.14",
        "core_version": "1.2.1+55.g590fe.HEAD"
      },
      "signature": {
        "key_id": "95f5a194-b0a4-4351-a",
        "checksum": "AF09217C56944D91C47847A77C0367C594C0B948E7E75654B889571BD4F60A71"
      }
    }
```

Chapter 8

No Access Behavior

The security policy maintains a **No Access Operation**, configured in an ESA, which determines the response for unauthorized unprotect requests.



The following table describes the result returned in the response for the various no access unprotect permissions.

No Access Operation	Data Returned
Null	null
Protected	(protected value)
Exception	Query will fail with an exception

> **Note:**
> An unauthorized protect will throw an exception.

Chapter 9

Upgrading to the Latest Version

You can download the latest version of the deployment package from *https://my.protegrity.com*. Navigate to **Data Protection** > **Cloud Protect** to download the latest version.

After downloading the deployment package from the Protegrity Portal, unzip the package to extract the artifact files. In the AWS Console, navigate to the S3 bucket that was previously created to upload deployment artifacts (see: *Create S3 bucket for Installing Artifacts*).

> **Note:**
> Only extract the deployment package and not the files in it.

Upload the following artifacts to the S3 bucket:

- *protegrity-protect-<version>.zip*
- *protegrity-agent-<version>.zip*
- *protegrity-external-extension-<version>.zip*
- *protegrity-sample-policy-<version>.zip*

If the release version matches your existing deployment, you don't need to upload it again. Save the following artifacts on your local system so that you have them available during the next steps:

- *pty_protect_api_cf.json*
- *pty_agent_cf.json*
- *pty_log_forwarder_cf.json*

Perform the following steps to upgrade the **Agent Lambda**, **Log Forwarder Lambda** and **Protect Lambda** separately.

> **Important:**
> If new version is available for Agent Lambda, Agent Lambda must be upgraded first.

9.1 Disable Protegrity Agent Function CloudWatch Event Rule

Cloud Watch Event Rule is used to periodically run Protegrity Agent Function to synchronize policy from ESA. This functionality is optional when deploying Protegrity Serverless Solution. If the Event Rule is enabled, it must be disabled temporarily for the time of the upgrade process.

Follow the steps below to determine if your deployment uses Event Rule and disable it.

1. Go to AWS Cloud Formation and select existing Protegrity deployment stack.
2. Select **Resources** tab from the top portion of the screen.
3. Check if there is a resource with **ScheduledRule** LogicalID. If there is no such resource you can skip to **Upgrading Policy Agent Lambda** section. If the scheduled rule is there, continue with the next steps in this section.
4. Click on the **Physical ID** link in the **ScheduledRule** row. The link opens Policy Agent Event Rule configuration.
5. Select **Disable** from the top-right portion of the screen. This will disable the rule. You will re-enable it after the upgrade process is complete.

9.2 Upgrading Policy Agent Lambda

> **Note:**
> If the release version of the artifact zip file has not changed since the previous installation, you can skip the Agent Lambda upgrade.

1. Go to AWS Lambda console and select existing **Protegrity Agent Lambda**.
2. Click **Actions** in top right portion of the screen. Select **Publish new version**. Click **Publish**. The version of Agent Lambda you just created will serve as restore point in the case you needed to rollback the upgrade.
3. Go to Lambda **Configuration** > **Environment variables**.
4. Record environment variables values. You will use them later to configure upgraded Lambda Function. You can use the aws cli command below to save the function variables into the local json file:

```
aws lambda get-function-configuration --function-name \
arn:aws:lambda:<aws_region>:<aws_account>:function:<function_name> \
--query Environment > <function_name>_env_config.json
```

5. Go to **AWS Cloud Formation** and select existing Protegrity Agent deployment stack.
6. Select **Update**. Check **Replace current template** > **Upload a template file**.
7. Upload *pty_agent_cf.json* file and select **Next**.
8. Click **Next** until **Review** window and then select **Update stack**.
9. Wait for the **Cloud Formation** to complete.
10. Navigate back to Agent Lambda Function.

> **Note:** Make sure you are viewing the latest Lambda Function, not the published version.

11. Go to **Configuration** > **Environment variables**. Replace placeholder values with values recorded in previous step. Alternatively, you can run the following aws cli command to update function configuration using json file saved in the previous steps:

```
aws lambda update-function-configuration --function-name \
arn:aws:lambda:<aws_region>:<aws_account>:function:<function_name> \
--environment file://./<function_name>_env_config.json
```

> **Note:** If your current agent installation version is lower than 3.0.12, make sure you set the following function configuration variables:
>
> - PTY_ADDIPADDRESSHEADER
> - PTY_ESA_CA_SERVER_CERT
>
> You can read more about these variables in section *Policy Agent Lambda Configuration*.

12. If you are upgrading to version 1.6.1 or higher please follow the steps below, otherwise the upgrade process is completed.
13. From AWS Console, navigate to **IAM** > **Policies**
14. Search for the Agent Lambda IAM Policy created in *Create Agent Lambda IAM policy*
15. Click on the policy, then select **Edit Policy**. Select **JSON** tab.
16. Add the following statement to the list of policy statements.

```
{
  "Sid": "LambdaGetConfiguration",
  "Effect": "Allow",
  "Action": [
      "lambda:GetFunctionConfiguration"
  ],
  "Resource": [
      "arn:aws:lambda:*:*:function:*"
  ]
}
```

17. Click **Review Policy**, then **Save Changes**. Wait for the changes to save.

9.3 Upgrading Log Forwarder Lambda

Publish Log Forwarder Lambda Version

Publishing a version of the Log Forwarder Lambda allows to roll-back to pre-existing version if upgrade fails

1. Go to AWS Lambda console and select existing **Protegrity Log Forwarder Lambda**.
2. Click **Actions** in top right portion of the screen. Select **Publish new version**. Click **Publish**.
3. Record the Lambda version number. It will be displayed at the top of the screen. You can also retrieve it from the Lambda function view, under **Versions** tab.

   Log Forwarder Lambda version number for roll-backs: _____

Disable Kinesis Trigger

Disabling Kinesis trigger ensures there are no unprocessed or re-processed events while function is upgraded.

1. Go to AWS Lambda console and select existing **Protegrity Log Forwarder Lambda**.
2. Select **Configuration tab** > **Triggers**

3. Check **Kinesis** trigger and click **Edit** button
4. Uncheck **Activate trigger** and click **Save**
5. Wait for function to stop processing events by monitoring function in **Monitor** tab

Upgrade Forwarder Lambda Version

Upgrade Log Forwarder function with new code

1. Go to AWS Cloud Formation and select existing Protegrity Log Forwarder deployment stack.
2. Select **Update Stack** > **Make a direct update**.
3. Select **Replace existing template** > **Upload a template file**.
4. Upload *pty_log_forwarder_cf* file and select **Next**.
5. Click **Next** until **Review** window and then select **Update stack**.
6. Wait for the **Cloud Formation** to complete.

Enable Kinesis Trigger

1. Go to AWS Lambda console and select existing **Protegrity Log Forwarder Lambda**.
2. Select **Configuration tab** > **Triggers**
3. Check **Kinesis** trigger and click **Edit** button
4. Check **Activate trigger** and click **Save**

Log Forwarder function will now start processing events from where it left off when Kinesis trigger was disabled.

Monitor and roll-back

Monitor Log Forwarder function for errors in its CloudWatch logs and in **Montior** tab. To roll back function to the previous version if any errors occur follow these steps:

1. Go to AWS Lambda console and select existing **Protegrity Log Forwarder Lambda**.
2. Select **Configuration tab** > **Triggers**
3. Expand **Details** section of **Kinesis** trigger and record **UUID** value
4. Execute the following AWS CLI command to move Kinesis trigger to previous version of Log Forwarder Lambda that was created earlier and recorded as **Log Forwarder Lambda version number for roll-backs**. Substitute *kinesis-mapping-uuid*, *log-forwarder-function-name*, *version-for-roll-backs* with your values:

```
aws lambda update-event-source-mapping --uuid <kinesis-mapping-uuid> --function-name
<log-forwarder-function-name>:<version-for-roll-backs>
```

5. Find Kinesis trigger attached to previous version of Log Forwarder Lambda by navigating **Versions tab** > **Version number link in the Versions column**

Kinesis trigger is now moved to previous version of Log Forwarder Lambda function.

9.4 Upgrading Protect Lambda

> **Note:**
> If the release version of the artifact zip file has not changed since the previous installation, you can skip the Protect Lambda upgrade.

Diagram below illustrates upgrade steps.

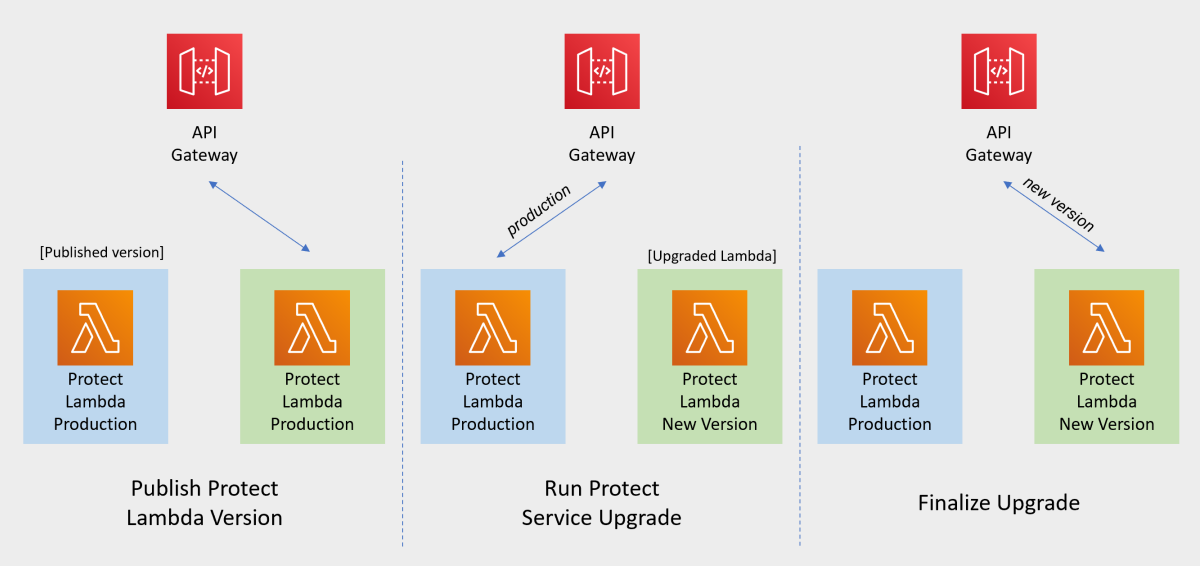


Figure 9-1: Protect Lambda Upgrade Steps

Publish Protect Lambda Version

Publishing a version of the Protect Lambda allows updating it without interruptions to the existing traffic.

1. Go to AWS Lambda console and select existing **Protegrity Protect Lambda**.
2. Go to Lambda **Configuration** > **Environment variables**.
3. Record environment variables values. You will use them later to configure upgraded Lambda Function. You can use the aws cli command below to save the function variables into the local json file:

   ```
   aws lambda get-function-configuration --function-name \
   arn:aws:lambda:<aws_region>:<aws_account>:function:<function_name> \
   --query Environment > <function_name>_env_config.json
   ```

4. Click **Actions** in top right portion of the screen. Select **Publish new version**. Click **Publish**.
5. Record the Lambda version number. It will be displayed at the top of the screen. You can also retrieve it from the Lambda function view, under **Versions** tab.

   Protect Lambda version number: _____

Run Protect Service Upgrade

In this step, the Protect service including Lambda $LATEST version will be updated using Cloud Formation template. The Lambda version created in previous step will be used to serve existing traffic during the upgrade process.

1. Go to AWS Cloud Formation and select existing Protegrity deployment stack.
2. Select **Update**. Check **Replace current template** > **Upload a template file**.
3. Upload *pty_protect_api_cf.json* file and select **Next**.
4. Update **ProtectFunctionProductionVersion** parameter with **Protect Lambda version number** recorded in step 3.
5. Click **Next** until **Review** window and then select **Update stack**.
6. Wait for the **Cloud Formation** to complete.
7. Go back to Lambda console and select Protect Lambda.

8. Go to **Configuration** > **Environment variables**. Replace placeholder values with values recorded in previous step. Alternatively, you can run the following aws cli command to update function configuration using json file saved in the previous steps:

```
aws lambda update-function-configuration --function-name \
arn:aws:lambda:<aws_region>:<aws_account>:function:<function_name> \
--environment file://./<function_name>_env_config.json
```

9. Navigate to **Aliases** tab. Verify that Production alias points to the lambda version you specified in the cloud formation template.
10. The upgraded Protect Lambda is configured with a sample policy. *Test function deployment*.
11. Run Agent Lambda Function.
12. The upgraded Protect Lambda is now configured with a Policy Agent policy. Test function deployment according to policy configuration.

Finalize upgrade

In this step, the Protect Lambda will be configured to serve traffic using $LATEST version upgraded in the previous step.

1. Go back to Protegrity AWS Cloud Formation deployment stack.
2. Select **Update**. Check **Use Current template**.
3. Update **ProtectFunctionProductionVersion** parameter with the following value: **$LATEST**.
4. Click **Next** until **Review** window and then select **Update stack**.
5. Go back to Lambda console and select Protect Lambda.
6. From the Lambda console, verify that Latest alias points to **$LATEST** version.
7. Test your function to make sure it works as expected.
8. If you need to rollback to older version of Protect Lambda, you can re-run the cloud formation with **ProtectFunctionProductionVersion** parameter set to the previous version of Protect Lambda.

9.5 Re-enable Protegrity Agent Function CloudWatch Event Rule

If the Event Rule was disabled at the beginning of the upgrade process, you must re-enabled it. Follow the steps below to re-enable Policy Agent Event rule.

1. Go to the Protegrity Agent Cloud Formation Stack.
2. Select **Resources** tab from the top portion of the screen.
3. Click on the **Physical ID** link in the **ScheduledRule** row. The link opens Policy Agent Event Rule configuration.
4. Select **Enable** from the top-right portion of the screen. This will enable the rule. You will re-enable it after the upgrade process is complete.

Chapter 10

Known Limitations

Known product limitations:

- FPE is supported only for ASCII values.

Appendix

Appendices

Appendix A

Protection Methods

For more information about the protection methods supported by Protegrity, refer to the *Protegrity Protection Methods Reference Guide*.

Table -1: Input Data Types Supported by Protegrity Tokenization

Tokenization Type	Supported Input Data Types	Notes
Numeric Credit Card Alpha Upper-case Alpha Alpha-Numeric Upper Alpha-Numeric Lower ASCII Printable Decimal Unicode Unicode Base64 Unicode Gen2 Email	STRING NULL	
Integer	NUMBER NULL	

Tokenization Type	Supported Input Data Types	Notes
Date Datetime	STRING NULL	For information about supported formats, refer to the *Protegrity Protection Methods Reference Guide*.
Binary	STRING NULL	Must be **`hex`** encoded unless a different **`encoding`** is specified. Another supported encoding is **`base64`**.

Table -2: Input Data Types Supported by No Encryption

Protection Method	Supported Input Data Types	Notes
No Encryption	STRING NUMBER NULL	

Table -3: Input Data Types Supported by Protegrity Encryption

Encryption Algorithm	Supported Input Data Types	Notes
3DES AES-128 AES-256 CUSP 3DES CUSP AES-128 CUSP AES-256	STRING	Must be **`hex`** encoded unless a different **`encoding`** is specified. Another supported encoding is **`base64`**.

Appendix B

ADFS Federation using AWS Cognito User Pools

Common use case for Protegrity Policy Member source integrates with customer's Active Directory. The AWS API Gateway can integrate with ADFS using AWS Cognito. The following article describes the procedure.

https://aws.amazon.com/blogs/mobile/building-adfs-federation-for-your-web-app-using-amazon-cognito-user-pools/

Appendix C

Invoke Lambda Directly

AWS Lambda can be invoked directly, such as from AWS SDK. This section contains information about request and response payloads with examples demonstrating direct invocation using AWS CLI and Python SDK (Boto3).

11.3.1 Request Payload

Lambda request payload for the direct invocation is defined as following

```
{
  "body": "<rest-api-request-payload>",
  "path": "/v1/<operation>",
  "headers": {}
}
```

- **body** - JSON string. Request schemas defined in *Rest API Request*.
- **path** - can be either '/v1/protect' or '/v1/unprotect'.
- **headers** - can be used to pass authorization headers. See example below.

Example request:

```
{
  "body": "{\"query_id\": \"3\",\"user\": \"user1\",\"data_element\": \"deAlpha\",\"data\":
[\"data1\", \"data2\"]}",
  "path": "/v1/protect",
  "headers": {}
}
```

Example Request with JWT authorization:

> **Note:** Lambda Environment sample configuration:
>
> ```
> authorization="jwt"
> jwt_verify=1
> allow_assume_user=1
> jwt_user_claim="username"
> jwt_secret_base64="Must be set to public certificate"
> ```
>
> See *REST API Authorization* for JWT configuration details.

```
{
  "body": "{\"query_id\": \"3\",\"user\": \"user1\",\"data_element\": \"deAlpha\",\"data\":
[\"data1\", \"data2\"]}",
  "path": "/v1/protect",
  "headers": {
    "authorization": "bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpYXQiOjE2MTM4NjIzNzEsImlkIjoiYS1iLWMtZC1lLWYtMS0yLTMiL
CJ1c2VybmFtZSI6IlBhdWwgQXRyZWlkZXMifQ.R1NcJ43540HKdhEBOK9WaMMpjBOYSJetckQKrcPQdz0z6sx1EDwHXYng
BP9DtHgUM-6Vf1VNjtFh_Nqfeepp1BavmigIXoe3ZbrxRI3DFKi2UuLmgn--
EYrSGlWsQjnmjaz5qUkID9iY2MtsRunKSuolSvG9UsD1G32kv0KZYX0"
  }
}
```

11.3.2 Response Payload

Lambda response payload has the following structure

```
{
  "body": "<rest-api-response-payload>"
  "isBase64Encoded": false,
  "statusCode": <http-status-code>,
}
```

- **body** - JSON string. Response schemas defined in *Rest API Response*.
- **isBase64Encoded** - always set to false.
- **statusCode** - HTTP status code integer.

Success Response Payload Example:

```
{
  "body": "{\"encoding\":\"utf8\",\"results\":[\"xcgd\", \"migs\"],\"success\":true}",
  "isBase64Encoded": false,
  "statusCode": 200
}
```

11.3.3 Error Response

Cloud API Lambda returns following error responses depending on the error type

Cloud API Protection Operation Error
Returned when invalid data element is used or user has insufficient permissions to execute security operation.

```
{
  "body": "{\"error_msg\":\"Unprotect failed. Data element not found. Refer to audit log for
details.\",\"success\":false}",
  "isBase64Encoded": false,
  "statusCode": 400
}
```

Cloud API Invalid Request Error
Missing fields in the incoming request or malformed request JSON.

```
{
  "body": "Request format is not supported",
  "isBase64Encoded": false,
  "statusCode": 400
}
```

Cloud API Unexpected Lambda Exception Error
Caused by Lambda runtime exception, for instance due to too short timeout or not enough memory.

```
{
  "errorMessage": "2023-01-18T16:42:19.593Z d0cf62d0-9eaf-427b-8ca5-1bdd8bd0b082 Task timed
out after 10.25 seconds"
}
```

11.3.4 Examples

Prerequisites:

- AWS SDK or Command Line
- AWS Access Key ID and AWS Access Key

> **Note:** IAM roles must follow the Principle of Least Privilege. Only users / services who must have access need to have invoke permissions on AWS Lambda.

See *Request Payload* for request payload examples.

AWS CLI command to invoke Cloud API Lambda function:

```
aws lambda invoke --function-name Protegrity_Protect_RESTAPI_{stackname} --payload
    fileb://request_payload.json --log-type Tail output
```

Sample Python code demonstrating Cloud API Direct Lambda Calls

```
import json
import logging
import boto3
lambda_client = boto3.client("lambda")
logging.basicConfig(format="%(message)s")
logger = logging.getLogger('pty_cloud_api_sample')
logger.setLevel(logging.DEBUG)


class ProtectClient(object):
  """
    Sample client demonstrating how to invoke Protegrity Cloud API Lambda

      protect_fn: str - Name of the Cloud API Lambda (for example,
Protegrity_Protect_RESTAPI_my_deployment)
  """

  def __init__(self, protect_fn):
    self.protect_fn = protect_fn

  def invoke_protect(self, values, data_element, operation, user, query_id,
             column_info=""):
    """
    Invokes Protegrity Cloud API Lambda to execute protect or unprotect operation

      values: list[str] - List of values to be protected/unprotected
      data_element: str - Name of the policy data element to use with protect/unprotect
operation
      operation: str - Either 'protect' or 'unprotect'
      user: str - Policy user
      query_id: str - Query id will be present in the audit log
      column_info: - Used for troubleshooting, for instance, when protecting values/rows from
multiple database columns
    """

    # Set authorization header here if JWT authorization is
    # enabled in Cloud API Function configuration
    headers = {"Authorization": ""}
    request_body = {
      "user": user,
      "data_element": data_element,
      "data": values,
      "query_id": query_id
    }
    payload = json.dumps({"body": json.dumps(request_body), "path": f"/v1/{operation}",
                  "headers": headers})
    logger.debug(f"Request payload: {payload}")
    response = lambda_client.invoke(FunctionName=self.protect_fn, Payload=payload)
    lambda_response_payload = json.loads(response["Payload"].read().decode())
    logger.debug(f"Response payload: {lambda_response_payload}")
    response_status_code = lambda_response_payload.get("statusCode")
    response_body_string = lambda_response_payload.get("body")
    if response_status_code == None or response_body_string == None:
      raise Exception(f"Unexpected Cloud API Lambda error: [{lambda_response_payload}]")
    try:
      body_json = json.loads(response_body_string)
      if response_status_code == 200:
        return body_json.get("results", [])
      elif body_json.get("error_msg"):
        raise Exception(f"Cloud API Lambda error: [{response_status_code} -
{body_json.get('error_msg')}]")
      raise Exception(f"Unexpected Cloud API Lambda error: [{lambda_response_payload}]")
    except json.decoder.JSONDecodeError:
      # Cloud API may return error in the response body
      # For example, {"statusCode": 400, "body":"Error message"}
      raise Exception(f"Cloud API Lambda error: [{response_status_code} -
{response_body_string}]")

# Replace cloud-api-lambda-name with the name of the Cloud API Lambda
# For example, Protegrity_Protect_RESTAPI_my_deployment
protect_client = ProtectClient('cloud-api-lambda-name')
```

```
protected_data = ["UtfVk UHgcD!"]
logger.info(f"Protected data: {protected_data}")
unprotected_data = protect_client.invoke_protect(
  values=protected_data,
  data_element='alpha',
  operation='unprrotect',
  user='test-user',
  query_id='1234')
logger.info(f"Unprotected data: {unprotected_data}")
```

Appendix D

Policy Agent - Custom VPC Endpoint Hostname Configuration

The Policy Agent uses default endpoint hostnames to communicate with other AWS services (for example, secretsmanager.amazonaws.com). This configuration will only work in VPCs where Amazon-provided DNS is available (default VPC configuration with private DNS option enabled for the endpoint). If your VPC uses custom DNS, follow the instructions below to configure the Policy Agent Lambda to use custom endpoint hostnames.

> **Note:**
> This configuration is only available with the Cloud Protect version 1.5.0 or higher. For more information about the upgrade instructions, refer to *Upgrading to the Latest Version*.

11.4.1 Identify DNS Hostnames

➤ To identify DNS hostnames:

1. From AWS console, select **VPC** > **Endpoints**.
2. Select Secrets Manager endpoint from the list of endpoints.
3. Under **Details** > **DNS Names**, note the private endpoint DNS names adding https:// at the beginning of the endpoint name. For example, *https://vpce-1234-4pzomrye.kms.us-west-1.vpce.amazonaws.com*
4. Note down DNS names for the KMS and Lambda endpoints:
   AWS_SECRETSMANAGER_ENDPOINT: https://_____
   AWS_KMS_ENDPOINT: https://_____
   AWS_LAMBDA_ENDPOINT: https://_____

11.4.2 Update the Policy Agent Lambda configuration

To update policy agent lambda configuration:

1. From the AWS console, navigate to Lambda, and select the Policy Agent Lambda function.
2. Select the **Configuration** section and choose **Environment variables**.
3. Select **Edit** and add the following environment variables with the corresponding endpoint URLs recorded in steps 3-4:

   Table -4:

Parameters	Value
AWS_SECRETSMANAGER_ENDPOINT_URL	<AWS_SECRETS_ENDPOINT>
AWS_KMS_ENDPOINT_URL	<AWS KMS ENDPOINT>
AWS_LAMBDA_ENDPOINT_URL	<AWS LAMBDA ENDPOINT>

4. Click **Save** and **Run** the Lambda. The Lambda will now use endpoints you have just configured.

Appendix E

Installing the Policy Agent and Protect Lambda in Different AWS Accounts.

The Policy Agent Lambda function and Protect Lambda functions can be installed in separate AWS accounts. However, additional configuration is required to authorize the Policy Agent to provision the security policy to a remote Protect Lambda function.

> **Note:** The Policy Agent will deploy an encrypted security policy file to an S3 bucket in the Protect function's AWS Account.

11.5.1 Create Agent Lambda IAM policy

1. Login to the AWS account that hosts the Protect Lambda function.
2. From the AWS IAM console, select **Policies** > **Create Policy**.
3. Select the JSON tab and copy the following snippet.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LambdaUpdateFunction",
      "Effect": "Allow",
      "Action": [
        "lambda:UpdateFunctionConfiguration"
      ],
      "Resource": [
        "arn:aws:lambda:*:*:function:*"
      ]
    },
    {
      "Sid": "LambdaReadLayerVersion",
```

```
      "Effect": "Allow",
      "Action": [
        "lambda:GetLayerVersion",
        "lambda:ListLayerVersions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "LambdaDeleteLayerVersion",
      "Effect": "Allow",
      "Action": "lambda:DeleteLayerVersion",
      "Resource": "arn:aws:lambda:*:*:layer:*:*"
    },
    {
      "Sid": "LambdaPublishLayerVersion",
      "Effect": "Allow",
      "Action": "lambda:PublishLayerVersion",
      "Resource": "arn:aws:lambda:*:*:layer:*"
    },
    {
      "Sid": "S3GetObject",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::*/*"
    },
    {
      "Sid": "S3PutObject",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::*/*"
    },
    {
      "Sid": "LambdaGetConfiguration",
      "Effect": "Allow",
      "Action": [
          "lambda:GetFunctionConfiguration"
      ],
      "Resource": [
          "arn:aws:lambda:*:*:function:*"
      ]
    }
  ]
}
```

4. Replace the wildcards**(*)** with the region, account, and resource name information where required.
5. Select **Review policy**, type in the policy name, and confirm. Record policy name:
   Agent Lambda Cross Account Policy Name: _____

11.5.2 Create Policy Agent cross-account IAM Role

1. Login to the AWS account that hosts the Protect Lambda function.
2. From the AWS IAM console, select **Roles** > **Create Role**
3. Select **AWS Service** > **Lambda** > **Next**.
4. Select Policy created in the step above.
5. Proceed to **Name, Review and Create**.
6. Type the role name, for example, *ProtegrityAgentCrossAccountRole* and click **Confirm**.
7. Select **Create role**.

8. Record the role ARN.

   Policy Agent Cross Account IAM Role Name: _____

11.5.3 Allow the Policy Agent Cross-Account Role to be Assumed by the Policy Agent IAM Role

1. Login to the AWS account that hosts the Protect Lambda function.
2. Navigate to the previously created IAM Role (**Agent Lambda Cross-Account IAM Role Name**).
3. Navigate to **Trust Relationships** > **Edit trust policy**.
4. Modify the Policy Document replacing the placeholder value indicated in the following snippet as **<Agent Lambda IAM Execution Role ARN>** with the value recorded in *Create Protect Lambda IAM Role*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",

   "Principal": {

            "AWS": "<Agent Lambda IAM Execution Role ARN>"

      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

5. Click **Update policy**.

11.5.4 Add Assume Role to the Policy Agent Execution IAM Role

1. Login to the AWS account that hosts the Policy Agent.
2. Search for Agent Lambda IAM Execution Role Name created in *Create Agent Lambda IAM policy*.
3. Add Inline Policy.
4. Modify the Policy Document replacing the placeholder value indicated in the following snippet as **<Agent Lambda Cross-Account IAM ARN>** with the value recorded in the previous step.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": "<Agent Lambda Cross-Account IAM  ARN>"
    }
  ]
}
```

5. Modify the Policy Document replacing the placeholder value indicated in the following snippet as **<Agent Lambda IAM Execution Role ARN>** with the value recorded in the previous step.
6. When you are finished, choose **Next**.
7. On the **Create policy** page, type a **Name**, then choose **Create Policy**.

11.5.5 Update the Policy Agent Lambda Configuration

1. From the AWS console, navigate to Lambda, and select the Policy Agent Lambda function.
2. Select **Configuration** tab | **Environment variables**.
3. Select Edit and add the following environment variables with the value from **Agent Lambda Cross-Account IAM ARN**:

Parameter	Value
AWS_ASSUME_ROLE	Agent Lambda Cross-Account IAM ARN

4. Ensure the values in the Parameters **AWS_POLICY_S3_BUCKET**, **AWS_PROTECT_FN_NAME** and **AWS_POLICY_LAYER_NAME** are all in the Protect Lambda Function AWS Account.
5. In case custom VPC hostname configuration is used, you will need to set the ENDPOINT_URL. Refer to *Policy Agent - Custom VPC Endpoint Hostname Configuration*.

AWS_VPC_ENDPOINT_URL	<AWS_VPC_ENDPOINT>

6. Click **Save** and **Run** the Lambda. The Lambda will now assume the Role in Protect Lambda Function AWS Account and update the policy cross accounts.

Appendix F

Configuring Regular Expression to Extract Policy Username

Cloud Protect Lambda Function exposes USERNAME_REGEX configuration to allow extraction of policy username from user in the request.

USERNAME_REGEX Lambda Environment configuration

The USERNAME_REGEX configuration can be used to extract policy username from user in the request. The following are allowed values for USERNAME_REGEX:

- 1 - Default build-in regular expression is used:

```
^arn:aws:(?:iam|sts)::[0-9]{12}:(?:role|user|group|assumed\-role|federated\-user)\/([\w\/
+=,.\-]{1,1024}|[\w\/+=,.\-@]{1,1024})(?:@[a-zA-Z0-9\-]{1,320}(?:\.\w+)+)?$
```

- ^User regex$ - Custom regex with one capturing group. This group is used to extract the username.

Examples below show different regular expression values and the resulting policy user.

USERNAME_REGEX	User in the request	Effective Policy User
Not Set	arn:aws:iam::123456789012:user/juliet.snow	arn:aws:iam::123456789012:user/juliet.snow
	arn:aws:sts::123456789012:assumed-role/TestSaml	arn:aws:sts::123456789012:assumed-role/TestSaml
1	arn:aws:iam::123456789012:user/juliet.snow	juliet.snow
	arn:aws:sts::123456789012:assumed-role/TestSaml	TestSaml
`^arn:aws:(?:iam|sts)::[0-9]{12}:((?:role|user|group|assumed-role|federated-user).*)$`	arn:aws:iam::123456789012:user/juliet.snow	user/juliet.snow
	arn:aws:sts::123456789012:assumed-role/TestSaml	assumed-role/TestSaml

Appendix G

Associating ESA Data Store With Cloud Protect Agent

ESA controls which policy is deployed to protector using concept of data store. A data store may contain a list of IP addresses identifying servers allowed to pull the policy associated with that specific data store. Data store may also be defined as default data store, which allows any server to pull the policy, provided it does not belong to any other data stores. Node registration occurs when the policy server (in this case the policy agent) makes a policy request to ESA, where the agent's IP address is identified by ESA.

> **Note:** For more information about ESA data store refer to *Policy Management Guide* which is part of Protegrity ESA documentation.

Policy agent lambda source IP address used for node registration on ESA depends on ESA hubcontroller configuration ASSIGN_DATASTORE_USING_NODE_IP and the PTY_ADDIPADDRESSHEADER configuration exposed by the agent lambda.

The Lambda service uses multiple network interfaces, internal network interface with ephemeral IP range of 169.254.x.x and external network interface with IP range of the VPC subnet the Lambda is associated with. By default, when agent lambda is contacting ESA to register node for policy download, ESA uses agent Lambda VPC IP address. This default behavior is caused by the default ESA hubcontroller configuration ASSIGN_DATASTORE_USING_NODE_IP=false and agent default configuration PTY_ADDIPADDRESSHEADER=yes.

In some cases, when there is a proxy server between the ESA and agent lambda, the desirable ESA configuration is ASSIGN_DATASTORE_USING_NODE_IP=true. and PTY_ADDIPADDRESSHEADER=no which will cause the ESA to use proxy server IP address.

The table below shows how the hubcontroller and agent settings will affect node IP registration on ESA.

Table -5: Policy agent Lambda source IP address used for node registration on ESA

Agent source IP	Agent VPC subnet IP	Proxy IP	ESA config - ASSIGN_DATASTORE_USING_NODE_IP	Agent lambda config - PTY_ADDIPADDRESSHEADER	Agent node registration IP
169.254.144.81	10.1.2.173	No Proxy	true	yes	169.254.144.81
			true	no	10.1.2.173
			false	yes	
			false	no	
169.254.144.81	10.1.2.173	34.230.42.110	true	yes	169.254.144.81
			true	no	34.230.42.110
			false	yes	
			false	no	