Shraddha Jain

**Abstract**

The goal of this document is to help develop a tagging strategy that enables you to manage your AWS resources more effectively.

**AWS RESOURCE TAGGING Enforcement**

Contents

[1. Objective 3](#_Toc79408192)

[2. Intended Audience 3](#_Toc79408193)

[3. Configuring AWS Organization 3](#_Toc79408194)

[3.1 Sequence of Steps: 4](#_Toc79408195)

[3.2 Prerequisites: 4](#_Toc79408196)

[3.3 Execution: 4](#_Toc79408197)

[Step 1: Setting Up AWS Organizations: 4](#_Toc79408198)

[Step 2: Adding a New Account/Inviting an Account 4](#_Toc79408199)

[4. Develop SCPs in AWS Organization 8](#_Toc79408200)

[4.1 Sequence of Steps: 9](#_Toc79408201)

[4.2 Prerequisites: 10](#_Toc79408202)

[4.3 Execution: 10](#_Toc79408203)

[5. Testing a Policy: 16](#_Toc79408204)

[5.1 Sequence of Steps 16](#_Toc79408205)

[5.2 Prerequisites: 16](#_Toc79408206)

[5.3 Execution: 16](#_Toc79408207)

[6 Extract Untagged Resources 23](#_Toc79408208)

[6.1 Develop IAM policies and roles for Lambda Execution 23](#_Toc79408209)

[6.1.1 Sequence of Steps: 23](#_Toc79408210)

[6.1.2 Prerequisites: 23](#_Toc79408211)

[6.1.3 Execution: 23](#_Toc79408212)

[Step1: Develop IAM policies and execution role for Lambda execution as below: 23](#_Toc79408213)

[Step2: Develop python code for Lambda using Boto3 SDK. 24](#_Toc79408214)

[6.2 Email’s verification using SES 24](#_Toc79408215)

[6.2.1 Sequence of Steps: 24](#_Toc79408216)

[6.2.2 Prerequisites: 24](#_Toc79408217)

[6.2.3 Execution: 25](#_Toc79408218)

[6.3 Create role, lambda function and configure event rule to trigger Lambda function at regular frequency 26](#_Toc79408219)

[6.3.1 Sequence of Steps: 26](#_Toc79408220)

[6.3.2 Prerequisites: 26](#_Toc79408221)

[6.3.3 Execution: 26](#_Toc79408222)

[6.4 Validate Resources data CSV after Lambda execution at scheduled time 28](#_Toc79408223)

[6.4.1 Sequence of Steps: 28](#_Toc79408224)

[6.4.2 Prerequisites: 28](#_Toc79408225)

[6.4.3 Execution 28](#_Toc79408226)

Document Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision | Author | Email | Date | Status and Description |
| 0.1  0.2 | Sanchali Gupta  Shraddha Jain | [sanchali.gupta@nagarro.com](mailto:sanchali.gupta@nagarro.com)  shraddha.jain@nagarro.com | 30th July 2021 | First draft |
|  |  |  |  |  |

Document Approval History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision | Approver | Email | Date | Status |
| 1.0 | Pankaj Kumar | Pankaj.kumar04@nagarro.com |  |  |
|  |  |  |  |  |

# **1. Objective**

This document is for end-to-end automation of resource tagging enforcement and restrict resource creation without defined tags. The main purpose is to restrict the untagged resources and enforce resources to be tagged with the compliant tags mentioned in SCP policies. This can be done using AWS organization SCPs. Lambda function will make a report of untagged resources, not compliant tag resources and will notify with the help of AWS SES.

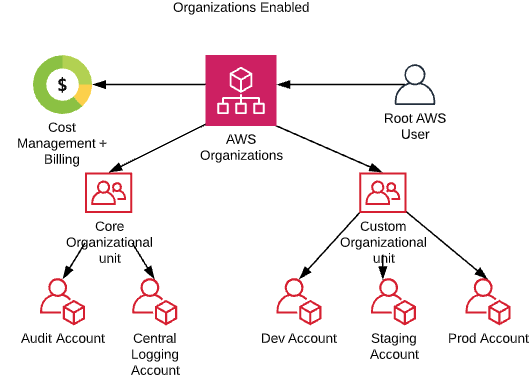
# **2. Intended Audience**

1. DevOps engineers.
2. Engineers working on AWS Cloud.
3. Organization Owner.

# **3. Configuring AWS Organization**

AWS Organizations helps you centrally govern your environment as you grow and scale your workloads on AWS. Whether you are a growing startup or a large enterprise, AWS Organizations helps you centrally manage billing; control access, compliance, and security; and share resources across your AWS accounts.

* Multi-account governance
* Networking and Service Control Policies
* Consolidated Billing
* Organizational Policies for Backups and Tagging



# **3.1 Sequence of Steps:**

* Setting Up AWS Organizations
* Adding a New Account/Inviting an Account

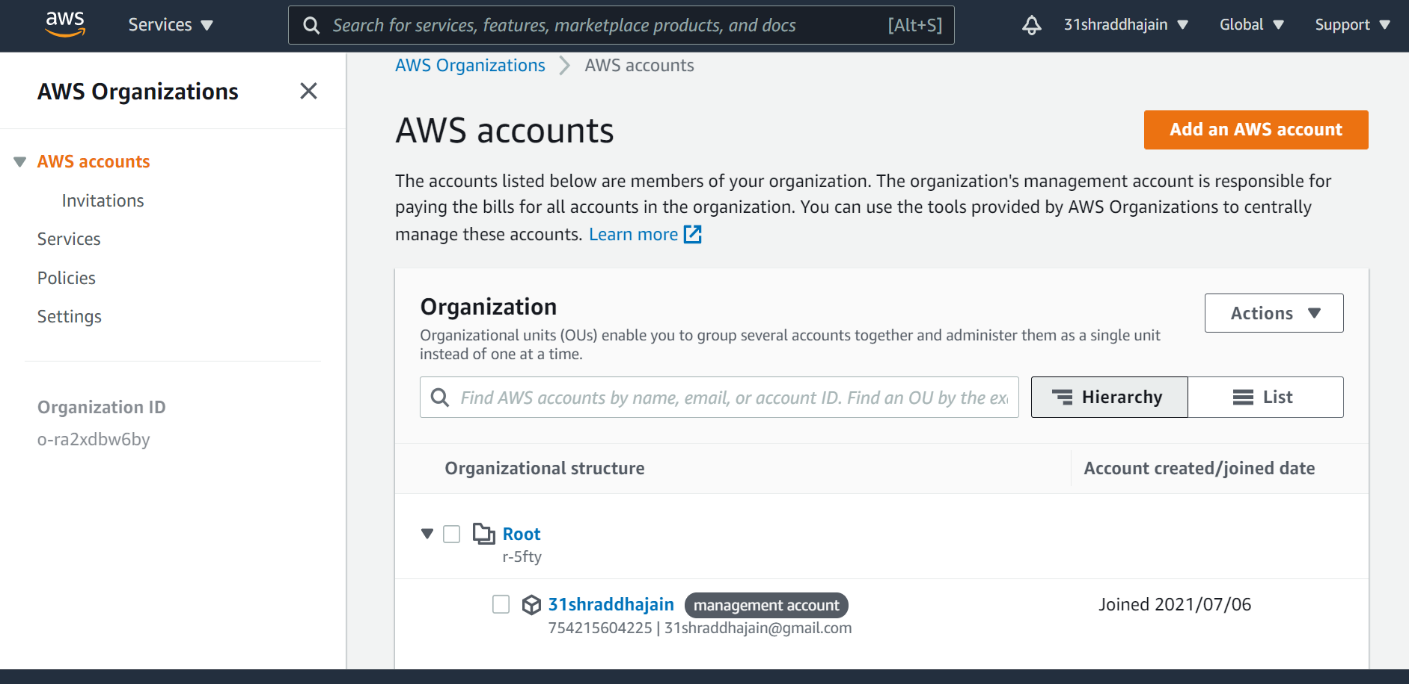
## **3.2 Prerequisites:**

AWS Organization at least one root account and one member account

# **3.3 Execution**:

## **Step 1: Setting Up AWS Organizations**:

In the console, navigate to the AWS Organizations page and click on the button to set up AWS Organizations.

Once the organization is set up, the view resembles this.

## **Step 2: Adding a New Account/Inviting an Account**

There are two ways of adding accounts to the organization. The first is to create a new account, and the second is to invite an existing account.

When we click on the “Add Account” item on the AWS Organizations dashboard, you are prompted to choose an option.

Graphical user interface, text, application, email

Description automatically generated

When inviting an account, select “**Invite an existing AWS account**”, which asks you to enter the account ID of the account you want to invite, and an opportunity to enter a message. Adding a message is important as the target account is notified of your request and must agree to join the organization.

Graphical user interface, text, application, email

Description automatically generated

The target account receives an email advising of the invitation or receives invitation in an AWS Organizations, and it is up to the owner of that account to accept or reject the invitation.

Graphical user interface, text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Once the invitation has been accepted and completed, the target account owner is notified via email, and their view in AWS organizations changes to show they are now part of the organization.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text

Description automatically generated

The master account view also changes to show the new accounts added to the organization.

Graphical user interface, text, application

Description automatically generated

# **4. Develop SCPs in AWS Organization**

Clicking on the Policies item in the navigation section displays the page where you can create, edit and delete policies, view their details and see the accounts and organizational units attached to the policy.

We can see a list of available policy services, click on “Service Control Policies” and enable the service.

Graphical user interface, text, application, email

Description automatically generated

**Service Control Policies (SCPs):**

Service Control Policies (SCPs) are a way of restricting the actions that can be taken in an AWS account so that all IAM users and roles, and even the root user cannot perform them. This feature is part of AWS Organizations, and the SCPs are controlled by the Organization Master account.

When the organization is initially defined, there is only one policy associated with the accounts, “FullAWSAccess”.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedThis is the content of “FullAWSAccess” policy.

# **4.1 Sequence of Steps:**

* Create a new SCP policy which requires a tag on specified resources
* Attach SCP Policy to a target account.

## **4.2 Prerequisites:**

* AWS Organization
* Enable Features of AWS Organization

# **4.3 Execution:**

**Step 1: Create a SCP policy which requires a tag on specified resources**

To Create a new policy, click on the “Create Policy” button.

The create Console screen will show, this screen has the Policy Name, Description, and then a GUI for creating the policy.

Graphical user interface, text, application

Description automatically generated

AWS has included an auto insert from all services listed on the right side menu, to make it easy to find what you are looking for.

Graphical user interface, application

Description automatically generated

Enter a name and description for the policy, and then navigate through the service list on the right to find any service like ‘EC2’. After selecting EC2, a list of actions is displayed. If you want to deny specific actions, select those, but here for example we want to deny only “RunInstances” action of EC2 service.

Graphical user interface, text, application

Description automatically generated

Before we can submit the policy, we have to select the resources to apply the policy too.

To do this, click on Add button of “Add a resource”

Graphical user interface, text, application

Description automatically generated

Select Service “EC2” then select resource type from the drop-down list “Instance” and then enter the Resource ARN.

Click on “Add resource” Button

Graphical user interface, application

Description automatically generated

Now, we are going to set a Condition clause.

Click on “Add” Button of Condition

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Once we have defined everything in our SCP, the resulting JSON looks like

Graphical user interface, text, application

Description automatically generated

The following SCP prevents IAM users and roles in the affected accounts from creating certain resource types if the request doesn't include the specified tags.

Graphical user interface, application

Description automatically generated

**Step 2: Attach SCP policy to a target account.**

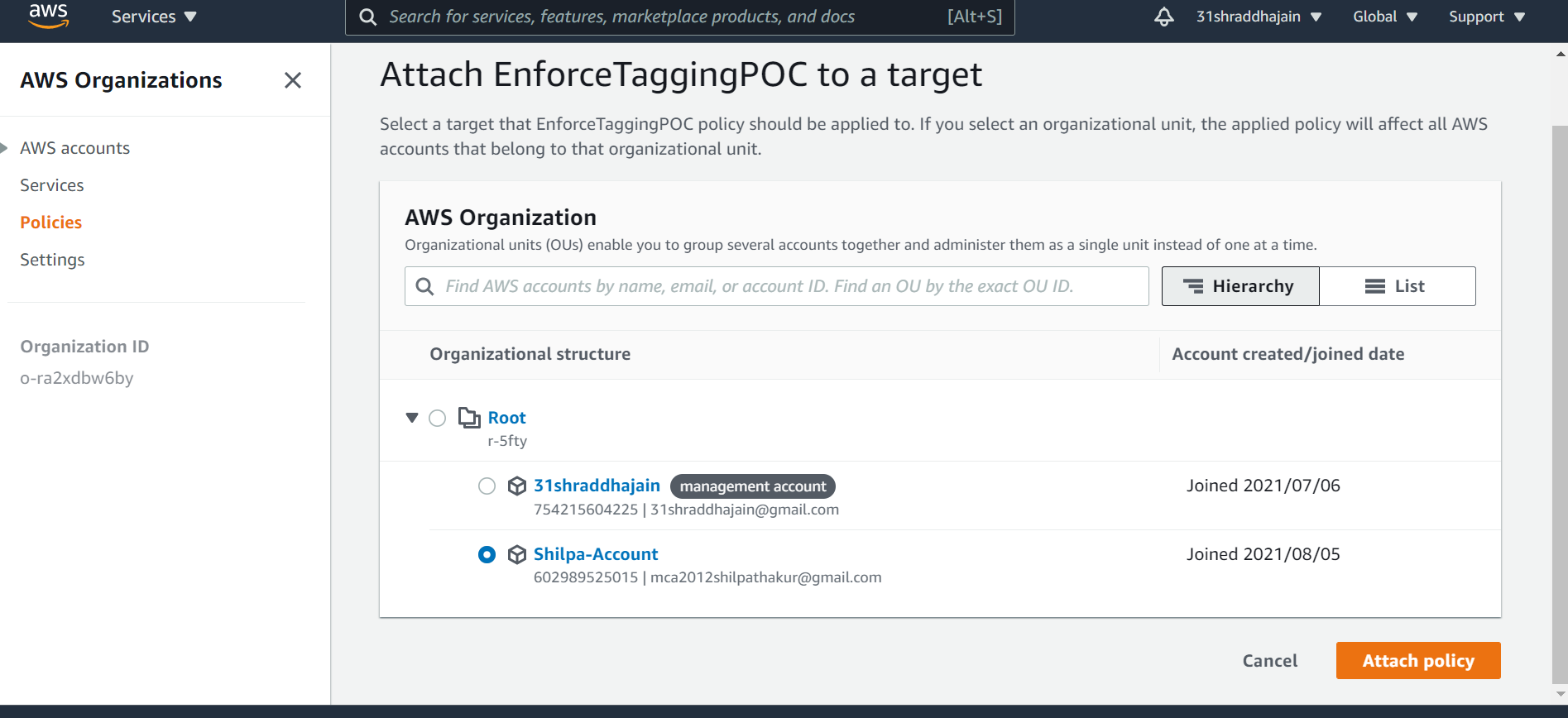
Service Control Policies are enabled, and we can add our policy to our target account by selecting the target account.

Select SCP Policy and then from actions select Attach policy

Graphical user interface, text, application, email

Description automatically generated

Select Target Account and then click on the attach policy button.



Now, we can see the target account attached to the SCP policy.

Graphical user interface, text, application, email

Description automatically generated

# **5. Testing a Policy:**

Let’s validate that the member account has this policy in effect or not. We do this by logging into the member or target account.

## **5.1 Sequence of Steps**

* Launch a EC2 Instance.

## **5.2 Prerequisites:**

* Target Account
* Service Access Rights

## **5.3 Execution:**

Let’s go to launch a EC2 Instance. Click on Launch Instances

Graphical user interface, text, application, email

Description automatically generated

Select an Amazon Machine Image (AMI)

Graphical user interface, text, application, email

Description automatically generated

After Selecting AMI, Choose Instance Type.

Table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Add Storage then click on Next button

Graphical user interface, text, email

Description automatically generated

Now, we can see there are no tags.

Graphical user interface, text, chat or text message, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

When we click on the “Launch Instances” button, we get a Launched Failed Error because the Service Control Policy is in place.

Graphical user interface, text, application, Word

Description automatically generated

Text

Description automatically generated

Again, trying to launch instance with Owner tag

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

In the same way, we can try for other services as well.

We have done it for around 25-30 services.

EC2, SNS, EFS, ELB, ASG, VPC, IAM, SQS, KMS, CloudFormation, Secret Manager, Code Commit, Code Build, ResourceGroups, Cognito (User Pool), Elastic Cache, ElasticBeanstalk, DMS, RDS, ECR, CloudWatch, CloudFront, WAF (Web ACL), Code Deploy, SSM, ECS.

The SCPs source code is available in git hub repo under:

[aws-resource-tagging-poc-work/DenyPossibleResources.json at master · pkuma343/aws-resource-tagging-poc-work · GitHub](https://github.com/pkuma343/aws-resource-tagging-poc-work/blob/master/first_part/DenyPossibleResources.json)

# **6 Extract Untagged Resources**

The main objective is to track down all the resources running across all regions of AWS account which are not having specified tag associated. A report containing all the resources running in AWS account which are not having specified tags associated gets created at regular intervals and sent to stakeholders. There is a Lambda function in place which utilizes the AWS API to generate a report.

# **6.1 Develop IAM policies and roles for Lambda Execution**

# **6.1.1 Sequence of Steps:**

* Develop IAM policies (trust and permission policy) json files for Lambda execution.
* Develop python code for Lambda using Boto3 SDK.

## **6.1.2 Prerequisites:**

AWS account.

# **6.1.3 Execution:**

## **Step1: Develop IAM policies and execution role for Lambda execution as below:**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

The code can be referred under GitHub repository at [aws-resource-tagging-poc-work/extractResources/shellScript/policies at master · pkuma343/aws-resource-tagging-poc-work · GitHub](https://github.com/pkuma343/aws-resource-tagging-poc-work/tree/master/extractResources/shellScript/policies).

## **Step2: Develop python code for Lambda using Boto3 SDK.**

This uses python boto3 SDK and AWS ResourceGroupsTaggingAPI to extract list of all resources across region in an AWS account which are not having specified tags associated. It creates a CSV document containing information around Resource type, Region, ARN and Tagging.

It uses AWS SES service to send an email to required stakeholders.

The code can be referred under GitHub repository at [aws-resource-tagging-poc-work/lambda\_handler.py at master · pkuma343/aws-resource-tagging-poc-work · GitHub](https://github.com/pkuma343/aws-resource-tagging-poc-work/blob/master/extractResources/shellScript/lambda_handler.py).

# **6.2 Email’s verification using SES**

### **6.2.1 Sequence of Steps:**

1. Verify emails using AWS SES.

## **6.2.2** **Prerequisites**:

AWS account.

## **6.2.3** **Execution**:

**Step 1: Verify emails using AWS SES.**

A python script using BOTO3 initiates verification process for required email ids.

The code can be referred under GitHub repository at [aws-resource-tagging-poc-work/verify\_email\_identity.py at master · pkuma343/aws-resource-tagging-poc-work · GitHub](https://github.com/pkuma343/aws-resource-tagging-poc-work/blob/master/extractResources/shellScript/verify_email_identity.py).

Graphical user interface, text, email, website

Description automatically generated

Graphical user interface, text, website

Description automatically generated

# **6.3 Create role, lambda function and configure event rule to trigger Lambda function at regular frequency**

### **6.3.1 Sequence of Steps:**

1. Create lambda execution role using policies json files developed in previous steps.
2. Create lambda function using code developed in previous steps.
3. Create event rule to trigger Lambda function at regular frequency.

## **6.3.2 Prerequisites:**

AWS account

Email’s verification using SES (6.2)

## **6.3.3 Execution:**

This can be achieved using two different ways:

1. Shell Script
2. Terraform

**Step 1: Scripts**

**Shell Script**

Execute lambdaSetup.sh from local system/EC2 instance/CI-CD pipeline. When running a shell script, **$1** first argument is the IAM Role Name, **$2** Second argument is the Lambda Function Name, and **$3** Third argument is the Event-Rule Name.

Example: lambdaSetup.sh lambdaGetResourcesIAM lambdaGetResourcesFunc lambdaGetResourcesRule.

The code can be referred under GitHub repository at [aws-resource-tagging-poc-work/lambdaSetup.sh at master · pkuma343/aws-resource-tagging-poc-work · GitHub](https://github.com/pkuma343/aws-resource-tagging-poc-work/blob/master/extractResources/shellScript/lambdaSetup.sh)

OR

**Terraform**

Execute main.tf from local system/EC2 instance/CI-CD pipeline.

The code can be referred under GitHub repository at [aws-resource-tagging-poc-work/extractResources/terraformExtractRes at master · pkuma343/aws-resource-tagging-poc-work · GitHub](https://github.com/pkuma343/aws-resource-tagging-poc-work/tree/master/extractResources/terraformExtractRes)

**Step2: Validate Role, Lambda function and Event Rule.**

**Graphical user interface, text

Description automatically generatedIAM ROLE:**

**Lambda function:**

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text

Description automatically generated **Event Rule**:

# **6.4 Validate Resources data CSV after Lambda execution at scheduled time**

# **6.4.1** **Sequence of Steps:**

1. Check email and validate resources in CSV.

## **6.4.2** **Prerequisites:**

AWS account.

### **6.4.3 Execution**

Step1. Monitor lambda execution.

Step2. Check email and validate CSV data.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, table

Description automatically generated

**Benefits:**

1. Tags enable you to add customizable key-value pairs to resources.
2. You can configure tags to be displayed with resources and can search and filter by tag.
3. CSV File supports the ability to break down AWS costs by tag.
4. We can tag the untagged resources.

**Challenges:** There are some resources which cannot be tagged.

**Conclusion:** In this document we have seen how easily we can identify the untagged resources in AWS account by getting simple notification from Amazon Simple Email Service and can easily tag those resources or terminate them and for this we have used AWS Organization’s SCP policies and lambda function for code part and configure continuous trigger.

**References:**

1. <https://aws.amazon.com/blogs/aws-cost-management/cost-allocation-blog-series-3-enforce-and-validate-aws-resource-tags/>
2. <https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html>
3. <https://docs.aws.amazon.com/resourcegroupstagging/latest/APIReference/API_GetResources.html>

**End of the Document**