**CREATING A NEW POLICY**

To create a new IAM policy in API Gateway the steps to be followed are

**Step 1:**

When the below url is invoked, the list of apis in API Gateway will be shown.

[**URL**: <<protocol>>://<<host>>:<<port>>/rest/apigateway/apis](URL:%3c%3cprotocol%3e%3e://%3c%3chost%3e%3e:%3c%3cport%3e%3e/rest/apigateway/apis)

**Method**: GET

**Request Headers:**

Accept: application/json

Content-type: application/json

**Response Payload**

{

"apiResponse": [

{

"api": {

"apiName": "API Gateway Alias Management Service",

"apiVersion": "10.4",

"apiDescription": "API Gateway Alias Management Service allows you to manage aliases in the API Gateway. Any user with the 'Manage aliases' functional previlige can manage the aliases in API Gateway. By default, the users who are part of API-Gateway-Administrators group will have this privilege. \n\n An alias in API Gateway holds environment-specific property values that can be used in policy routing configurations. The aliases can be referred to in routing endpoints, routing rules, endpoint connection properties, outbound authentication tokens, and outbound HTTP headers instead of providing an actual value. The corresponding alias value based on the stage is obtained and is substituted in place of an alias name during run-time. Thus the same alias can be referred to in multiple policies and the change in a particular alias would affect all the policy properties in which it is being referred. \n\n There are mainly six types of alias \n 1. Simple alias - Holds a simple value \n 2. Endpoint alias - Holds the endpoint value along with additional properties such as connection timeout, read timeout, whether to pass security headers or not, keystore alias, key alias \n 3. HTTP Transport security alias - Contains the transport level security information required while accessing the native API. Used in the outbound policies \n 4. SOAP message security alias - Contains message level security information that is required to access the native API \n 5. webMethods Integration Server alias - Holds the ESB service value. Can be used in the Invoke webMethods IS policy for request and response processing.\n 6. Transformation alias - Holds a list of XSLT style sheets. Can be used in the XSLT Transformation policies for request and response processing.\n\n In addition to the above alias types, the configurations for \n SAML issuer - which holds the configuration to communicate to STS (Security Token Service) to retrieve the SAML token. Used in the Outbound policies, \n Authorization server - which holds configuration of authorization server(internal/external)for introspection/Dynamic client registration. Used in the token validation and OAuth2 client creation, \n Client metadata mapping - which holds the authorization service provider metadata information. Used in the Dynamic client registration for the authorization server alias and \n Service Registry- which holds registry endpoint and service paths information. Used in the client side and server side discovery \n are internally stored as aliases.",

"isActive": false,

"type": "REST",

"publishedPortals": [],

"systemVersion": 1,

"id": "02e782df-d281-4204-b489-80dd96b8dac7"

},

"responseStatus": "SUCCESS"

},

{

"api": {

"apiName": "API Gateway Application Management Service",

"apiVersion": "10.4",

"apiDescription": "API Gateway Application Management Service allows you to manage the Applications in API Gateway. Any user with the 'Manage applications' functional privilege can manage the APIs in the API Gateway. By default, the users who are part of either API-Gateway-Providers or API-Gateway-Administrators groups will have this privilege.\n\n An application defines the precise identifiers by which messages from a particular consumer application is recognized at run time. The identifiers can be, for example, user name in HTTP headers, a range of IP addresses, such that API Gateway can identify or authenticate the consumers that are requesting an API.\n\n. The ability of API Gateway to relate a request to a specific consumer application enables it to:\n\n 1. Control access to an API at run time (that is, allow only authorized consumer applications to invoke an API).\n\n 2. Monitor an API for violations of a Service-Level Agreement (SLA) for a specified application.\n\n 3. Indicate the consumer application to which a logged transaction event belongs.\n\n\n\n An application has the following attributes for specifying the identifiers:\n\n 1. IP address, which specifies one or more IP addresses that identify requests from a particular consumer application. Example: 192.168.0.10\n\n This attribute is queried when the Identify and Authorize Application policy is configured to identify consumer applications by IP address.\n\n 2. JWT claims set, which specifies one or more claims that identify requests from a particular consumer application. The claims are a set of name-value pairs that provide sufficient information about the application. Example: sub = Administrator.\n\n This attribute is queried when the Identify and Authorize Application policy is configured to identify consumer applications by JSON Web Token (JWT).\n\n 3. Consumer certificate, which specifies the X.509 certificates that identify requests from a particular consumer.\n\n This attribute is queried when the Identify and authenticate consumer policy is configured to identify the consumer applications by a consumer certificate.\n\n 4. Identification token, which specifies the host names, user names or other distinguishing strings that identify requests from a particular consumer application.\n\n This attribute is queried when the Identify and authenticate consumer policy action is configured to identify consumer applications by host name, HTTP user name, and WSS user name.\n\n \n\n As an API provider or an API Gateway Administrator you can create and manage applications, and register applications with the APIs.\n\n These are the high level stages of managing and using an application:\n\n 1. API developers request the API Gateway administrators to create an application for access as per the required identification criteria.\n\n 2. API Gateway provider or administrator validates the request and creates a new application, there by provisioning the application specific access tokens (API access key and OAuth credentials).\n\n 3. API Developer, upon finding a suitable API, sends a request to API Gateway for consumption by providing the application details.\n\n 4. After validating the request, API Gateway provider or administrator associates the application with the API. Keys are generated for applications and not for every API that the application consumes. \n\n Note: The approval process, if any, is handled by the requesting application and not handled by API Gateway.\n\n 5. The API developer can then use the application with the proper identifier (such as the access key or identifier) to access the API. \n\n Strategies: \n\n A strategy is a way to authenticate the incoming request and provides multiple authentication mechanisms or multiple authorization servers for a single authentication scheme. API Gateway identifies the application and validates the token submitted through the strategy configured in the application. You can create multiple strategies authorized by an API for an application. For example, in case of OAuth authentication scheme, you want the application to support both OKTA and PINGFederate or OKTA with multiple tenants. This can be configured as OAuth strategy for the application.",

"isActive": false,

"type": "REST",

"publishedPortals": [],

"systemVersion": 1,

"id": "2f8d2bc6-d73b-4357-b76d-af74620b4b64"

},

"responseStatus": "SUCCESS"

},

{

"api": {

"apiName": "MessengerApi",

"apiVersion": "1.0",

"isActive": true,

"type": "REST",

"publishedPortals": [],

"systemVersion": 1,

"id": "72e18d0b-2800-4e67-950b-5c81b57c97ee"

},

"responseStatus": "SUCCESS"

} ] }

**Step 2:**

When the below url is invoked, the details of the specified API will be shown.

URL: <<protocol>>://<<host>>:<<port>>/rest/apigateway/id of the specified API.

**Method**: GET

**Request Headers:**

Accept: application/json

Content-type: application/json

**Response Payload:**

{

"apiResponse": {

"api": {

"apiDefinition": {

"type": "rest",

"info": {

"version": "1.0",

"title": "MessengerApi"

},

"serviceRegistryDisplayName": "API\_Gateway\_Messenger\_Service",

"schemes": [],

"security": [],

"paths": {

"/messages": {

"get": {

"description": "The method retrieves a list of available employees in API Gateway.\n",

"operationId": "getAllMessages",

"responses": {

"200": {

"description": "This status code indicates that a list of the available employees are retrieved successfully.\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

},

"401": {

"description": "This status code indicates that either user didn't provide right credentials or user doesn't have required privileges to access this API.\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

}

},

"mockedResponses": {},

"mockedConditionsBasedCustomResponsesList": [],

"enabled": true,

"scopes": [],

"requestBody": {

"content": {},

"name": "getAllMessages"

}

},

"post": {

"description": "The method creates an employee details in API Gateway.\n",

"operationId": "addMessage",

"responses": {

"200": {

"description": "This status code indicates that the employee details has been successfully created in API Gateway.\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

},

"400": {

"description": "This status code indicates that an employee detail with the given ID already exists in API Gateway\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

},

"401": {

"description": "This status code indicates that either user didn't provide right credentials or user doesn't have required privileges to access this API.\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

}

},

"mockedResponses": {},

"mockedConditionsBasedCustomResponsesList": [],

"enabled": true,

"scopes": [],

"requestBody": {

"content": {},

"name": "addMessage"

}

},

"scopes": [],

"displayName": "crudrestfulAPI",

"enabled": true

},

"/messages/{messageID}": {

"get": {

"description": "The method retrieves the details of an employee in API Gateway.\n",

"operationId": "getMessage",

"responses": {

"200": {

"description": "This status code indicates that the details of the alias are successfully retrieved.\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

},

"401": {

"description": "This status code indicates that either user didn't provide right credentials or user doesn't have required privileges to access this API.\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

},

"404": {

"description": "This status code indicates that an employee details with the given id could not be found in API Gateway.\n",

"content": {},

"links": {},

"schema": {},

"examples": {},

"headers": {}

}

},

"mockedResponses": {},

"mockedConditionsBasedCustomResponsesList": [],

"enabled": true,

"scopes": [],

"requestBody": {

"content": {},

"name": "getMessage"

}

},

"put": {

"responses": {},

"mockedResponses": {},

"mockedConditionsBasedCustomResponsesList": [],

"enabled": true,

"scopes": [],

"requestBody": {

"content": {},

"name": "requestBody"

}

},

"delete": {

"responses": {},

"mockedResponses": {},

"mockedConditionsBasedCustomResponsesList": [],

"enabled": true,

"scopes": [],

"requestBody": {

"content": {},

"name": "requestBody"

}

},

"parameters": [

{

"in": "path",

"name": "messageID",

"parameterSchema": {

"type": "string"

},

"required": false,

"schema": {

"type": "string"

},

"type": "string"

}

],

"scopes": [],

"displayName": "/crudrestfulAPI1",

"enabled": true

}

},

"securityDefinitions": {},

"definitions": {},

"parameters": {},

"baseUriParameters": [],

"externalDocs": [],

"servers": [

{

"url": "http://localhost:8080/crudrestfulAPI/webapi",

"variables": {}

}

],

"components": {

"schemas": {},

"responses": {},

"parameters": {},

"examples": {},

"requestBodies": {},

"headers": {},

"links": {},

"callbacks": {}

}

},

"nativeEndpoint": [

{

"passSecurityHeaders": true,

"uri": "http://localhost:8080/crudrestfulAPI/webapi",

"connectionTimeoutDuration": 0,

"alias": false

}

],

"apiName": "MessengerApi",

"apiVersion": "1.0",

"maturityState": "Beta",

"isActive": true,

"type": "REST",

"owner": "Administrator",

"policies": [

"18821d71-5b36-46f0-bd26-c40bf8e5208a"

],

"scopes": [],

"publishedPortals": [],

"creationDate": "2019-03-19 14:00:57 GMT",

"lastModified": "2019-03-22 06:49:03 GMT",

"systemVersion": 1,

"id": "72e18d0b-2800-4e67-950b-5c81b57c97ee"

},

"responseStatus": "SUCCESS",

"gatewayEndPoints": [

"http://SAG-3HYB152:5555/gateway/MessengerApi/1.0"

],

"versions": [

{

"versionNumber": "1.0",

"apiId": "72e18d0b-2800-4e67-950b-5c81b57c97ee"

}

]

}

}

**Step 3:**

When this url is invoked, the details of the policies present in the specified API will be shown.

**URL**: <<protocol>>://<<host>>:<<port>>/rest/apigateway/policies/id of the policy

**Method**: GET

**Request Headers:**

Accept: application/json

Content-type: application/json

**Response Payload:**

{

"policy": {

"id": "18821d71-5b36-46f0-bd26-c40bf8e5208a",

"names": [

{

"value": "Default Policy for API MessengerApi",

"locale": "English"

}

],

"descriptions": [

{

"value": "",

"locale": "English"

}

],

"scope": {

"scopeConditions": []

},

"policyEnforcements": [

{

"enforcements": [

{

"enforcementObjectId": "a8400f80-e08f-4f4e-bdd9-9241e9a60cf9",

"order": null

}

],

"stageKey": "routing"

},

{

"enforcements": [

{

"enforcementObjectId": "0681c7f4-8711-44fa-8515-de78591f817e",

"order": null

}

],

"stageKey": "IAM"

},

{

"enforcements": [

{

"enforcementObjectId": "e2944292-e8ab-4b4e-a970-2285170e46d2",

"order": null

}

],

"stageKey": "transport"

}

],

"policyScope": "SERVICE",

"active": false,

"global": false,

"systemPolicy": false

}

}

**Step 4:**

I have already taken the generic structure of the policy what I have to add.

When I post that generic structure I'm getting the following response with a new id.

**URL**: <<protocol>>://<<host>>:<<port>>/rest/apigateway/policyActions

**Method**: POST

**Request Headers:**

Accept: application/json

Content-type: application/json

**Request Payload:**

{

"policyAction": {

"names": [

{

"value": "Identify & Authorize Application",

"locale": "en"

}

],

"templateKey": "evaluatePolicy",

"parameters": [

{

"templateKey": "logicalConnector",

"values": [

"OR"

]

},

{

"templateKey": "allowAnonymous",

"values": [

"false"

]

},

{

"templateKey": "IdentificationRule",

"parameters": [

{

"templateKey": "applicationLookup",

"values": [

"strict"

]

},

{

"templateKey": "identificationType",

"values": [

"apiKey"

]

}

]

},

{

"templateKey": "IdentificationRule",

"parameters": [

{

"templateKey": "applicationLookup",

"values": [

"strict"

]

},

{

"templateKey": "identificationType",

"values": [

"httpBasicAuth"

]

}

]

}

],

"active": false

}

}

**Response Payload:**

{

"policyAction": {

"id": "fb3d16a2-01c4-453e-9d8c-11d4355f0d4a",

"names": [

{

"value": "Identify & Authorize Application",

"locale": "en"

}

],

"templateKey": "evaluatePolicy",

"parameters": [

{

"templateKey": "logicalConnector",

"values": [

"OR"

]

},

{

"templateKey": "allowAnonymous",

"values": [

"false"

]

},

{

"templateKey": "IdentificationRule",

"parameters": [

{

"templateKey": "applicationLookup",

"values": [

"strict"

]

},

{

"templateKey": "identificationType",

"values": [

"apiKey"

]

}

]

},

{

"templateKey": "IdentificationRule",

"parameters": [

{

"templateKey": "applicationLookup",

"values": [

"strict"

]

},

{

"templateKey": "identificationType",

"values": [

"httpBasicAuth"

]

}

]

}

],

"active": false

}

}

**Step 5:**

In this step we have to add a new policy with a id (created in step 4) to a request payload. After performing put operation the policy will be activated in API Gateway.

**URL**: <<protocol>>://<<host>>:<<port>>/rest/apigateway/policies/ id of the policy

**Method:** PUT

**Request Headers:**

Accept: application/json

Content-type: application/json

**Request Payload:**

{

"policy": {

"id": "18821d71-5b36-46f0-bd26-c40bf8e5208a",

"names": [

{

"value": "Default Policy for API MessengerApi",

"locale": "English"

}

],

"descriptions": [

{

"value": "",

"locale": "English"

}

],

"scope": {

"scopeConditions": []

},

"policyEnforcements": [

{

"enforcements": [

{

"enforcementObjectId": "a8400f80-e08f-4f4e-bdd9-9241e9a60cf9",

"order": null

}

],

"stageKey": "routing"

},

{

"enforcements": [

{

"enforcementObjectId": "0681c7f4-8711-44fa-8515-de78591f817e",

"order": null

}

],

"stageKey": "IAM"

},

{

"enforcements": [

{

"enforcementObjectId": "e2944292-e8ab-4b4e-a970-2285170e46d2",

"order": null

}

],

"stageKey": "transport"

}

],

"policyScope": "SERVICE",

"active": false,

"global": false,

"systemPolicy": false

} }

**Response Payload:**

{

"policy": {

"id": "18821d71-5b36-46f0-bd26-c40bf8e5208a",

"names": [

{

"value": "Default Policy for API MessengerApi",

"locale": "English"

}

],

"descriptions": [

{

"value": "",

"locale": "English"

}

],

"scope": {

"scopeConditions": []

},

"policyEnforcements": [

{

"enforcements": [

{

"enforcementObjectId": "a8400f80-e08f-4f4e-bdd9-9241e9a60cf9",

"order": null

}

],

"stageKey": "routing"

},

{

"enforcements": [

{

"enforcementObjectId": "0681c7f4-8711-44fa-8515-de78591f817e",

"order": null

}

],

"stageKey": "IAM"

},

{

"enforcements": [

{

"enforcementObjectId": "e2944292-e8ab-4b4e-a970-2285170e46d2",

"order": null

}

],

"stageKey": "transport"

}

],

"policyScope": "SERVICE",

"active": false,

"global": false,

"systemPolicy": false

}

}

**Step 6:**

After all the steps done if we refresh our API Gateway, we can see that the policy which we requested is successfully created.