# Package 'paws.networking'

August 23, 2021

**Title** 'Amazon Web Services' Networking & Content Delivery Services

**Version** 0.1.12

**Description** Interface to 'Amazon Web Services' networking and content delivery services, including 'Route 53' Domain Name System service, 'CloudFront' content delivery, load balancing, and more <https://aws.amazon.com/>.

**License** Apache License (>= 2.0)

**URL** https://github.com/paws-r/paws

**BugReports** https://github.com/paws-r/paws/issues

**Imports** paws.common (>= 0.3.0)

**Suggests** testthat

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Collate** 'apigateway_service.R' 'apigateway_interfaces.R' 'apigateway_operations.R' 'apigatewaymanagementapi_service.R' 'apigatewaymanagementapi_interfaces.R' 'apigatewaymanagementapi_operations.R' 'apigatewayv2_service.R' 'apigatewayv2_interfaces.R' 'apigatewayv2_operations.R' 'appmesh_service.R' 'appmesh_interfaces.R' 'appmesh_operations.R' 'cloudfront_service.R' 'cloudfront_interfaces.R' 'cloudfront_operations.R' 'directconnect_service.R' 'directconnect_interfaces.R' 'directconnect_operations.R' 'elb_service.R' 'elb_interfaces.R' 'elb_operations.R' 'elbv2_service.R' 'elbv2_interfaces.R' 'elbv2_operations.R' 'globalaccelerator_service.R' 'globalaccelerator_interfaces.R' 'globalaccelerator_operations.R' 'route53_service.R' 'route53_interfaces.R' 'route53_operations.R' 'route53domains_service.R' 'route53domains_interfaces.R' 'route53domains_operations.R' 'route53resolver_service.R' 'route53resolver_interfaces.R' 'route53resolver_operations.R' 'servicediscovery_service.R' 'servicediscovery_interfaces.R' 'servicediscovery_operations.R'

**NeedsCompilation** no

**Author** David Kretch [aut, cre],
Adam Banker [aut],
Amazon.com, Inc. [cph]

**Maintainer** David Kretch <david.kretch@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-08-23 07:10:18 UTC

# R topics documented:



---

`apigateway` *Amazon API Gateway*

---

## Description

Amazon API Gateway helps developers deliver robust, secure, and scalable mobile and web application back ends. API Gateway allows developers to securely connect mobile and web applications to APIs that run on AWS Lambda, Amazon EC2, or other publicly addressable web services that are hosted outside of AWS.

## Usage

```
apigateway(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- apigateway(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| create_api_key | Create an ApiKey resource |
| create_authorizer | Adds a new Authorizer resource to an existing RestApi resource |
| create_base_path_mapping | Creates a new BasePathMapping resource |
| create_deployment | Creates a Deployment resource, which makes a specified RestApi callable over the internet |
| create_documentation_part | Create documentation part |
| create_documentation_version | Create documentation version |
| create_domain_name | Creates a new domain name |
| create_model | Adds a new Model resource to an existing RestApi resource |
| create_request_validator | Creates a ReqeustValidator of a given RestApi |
| create_resource | Creates a Resource resource |
| create_rest_api | Creates a new RestApi resource |
| create_stage | Creates a new Stage resource that references a pre-existing Deployment for the API |
| create_usage_plan | Creates a usage plan with the throttle and quota limits, as well as the associated API stages, |
| create_usage_plan_key | Creates a usage plan key for adding an existing API key to a usage plan |
| create_vpc_link | Creates a VPC link, under the caller's account in a selected region, in an asynchronous opera |
| delete_api_key | Deletes the ApiKey resource |
| delete_authorizer | Deletes an existing Authorizer resource |
| delete_base_path_mapping | Deletes the BasePathMapping resource |
| delete_client_certificate | Deletes the ClientCertificate resource |
| delete_deployment | Deletes a Deployment resource |
| delete_documentation_part | Delete documentation part |
| delete_documentation_version | Delete documentation version |
| delete_domain_name | Deletes the DomainName resource |
| delete_gateway_response | Clears any customization of a GatewayResponse of a specified response type on the given R |

| | |
|---|---|
| delete_integration | Represents a delete integration |
| delete_integration_response | Represents a delete integration response |
| delete_method | Deletes an existing Method resource |
| delete_method_response | Deletes an existing MethodResponse resource |
| delete_model | Deletes a model |
| delete_request_validator | Deletes a RequestValidator of a given RestApi |
| delete_resource | Deletes a Resource resource |
| delete_rest_api | Deletes the specified API |
| delete_stage | Deletes a Stage resource |
| delete_usage_plan | Deletes a usage plan of a given plan Id |
| delete_usage_plan_key | Deletes a usage plan key and remove the underlying API key from the associated usage plan |
| delete_vpc_link | Deletes an existing VpcLink of a specified identifier |
| flush_stage_authorizers_cache | Flushes all authorizer cache entries on a stage |
| flush_stage_cache | Flushes a stage's cache |
| generate_client_certificate | Generates a ClientCertificate resource |
| get_account | Gets information about the current Account resource |
| get_api_key | Gets information about the current ApiKey resource |
| get_api_keys | Gets information about the current ApiKeys resource |
| get_authorizer | Describe an existing Authorizer resource |
| get_authorizers | Describe an existing Authorizers resource |
| get_base_path_mapping | Describe a BasePathMapping resource |
| get_base_path_mappings | Represents a collection of BasePathMapping resources |
| get_client_certificate | Gets information about the current ClientCertificate resource |
| get_client_certificates | Gets a collection of ClientCertificate resources |
| get_deployment | Gets information about a Deployment resource |
| get_deployments | Gets information about a Deployments collection |
| get_documentation_part | Get documentation part |
| get_documentation_parts | Get documentation parts |
| get_documentation_version | Get documentation version |
| get_documentation_versions | Get documentation versions |
| get_domain_name | Represents a domain name that is contained in a simpler, more intuitive URL that can be cal |
| get_domain_names | Represents a collection of DomainName resources |
| get_export | Exports a deployed version of a RestApi in a specified format |
| get_gateway_response | Gets a GatewayResponse of a specified response type on the given RestApi |
| get_gateway_responses | Gets the GatewayResponses collection on the given RestApi |
| get_integration | Get the integration settings |
| get_integration_response | Represents a get integration response |
| get_method | Describe an existing Method resource |
| get_method_response | Describes a MethodResponse resource |
| get_model | Describes an existing model defined for a RestApi resource |
| get_models | Describes existing Models defined for a RestApi resource |
| get_model_template | Generates a sample mapping template that can be used to transform a payload into the struct |
| get_request_validator | Gets a RequestValidator of a given RestApi |
| get_request_validators | Gets the RequestValidators collection of a given RestApi |
| get_resource | Lists information about a resource |
| get_resources | Lists information about a collection of Resource resources |
| get_rest_api | Lists the RestApi resource in the collection |
| get_rest_apis | Lists the RestApis resources for your collection |

| | |
|---|---|
| get_sdk | Generates a client SDK for a RestApi and Stage |
| get_sdk_type | Get sdk type |
| get_sdk_types | Get sdk types |
| get_stage | Gets information about a Stage resource |
| get_stages | Gets information about one or more Stage resources |
| get_tags | Gets the Tags collection for a given resource |
| get_usage | Gets the usage data of a usage plan in a specified time interval |
| get_usage_plan | Gets a usage plan of a given plan identifier |
| get_usage_plan_key | Gets a usage plan key of a given key identifier |
| get_usage_plan_keys | Gets all the usage plan keys representing the API keys added to a specified usage plan |
| get_usage_plans | Gets all the usage plans of the caller's account |
| get_vpc_link | Gets a specified VPC link under the caller's account in a region |
| get_vpc_links | Gets the VpcLinks collection under the caller's account in a selected region |
| import_api_keys | Import API keys from an external source, such as a CSV-formatted file |
| import_documentation_parts | Import documentation parts |
| import_rest_api | A feature of the API Gateway control service for creating a new API from an external API d |
| put_gateway_response | Creates a customization of a GatewayResponse of a specified response type and status code |
| put_integration | Sets up a method's integration |
| put_integration_response | Represents a put integration |
| put_method | Add a method to an existing Resource resource |
| put_method_response | Adds a MethodResponse to an existing Method resource |
| put_rest_api | A feature of the API Gateway control service for updating an existing API with an input of e |
| tag_resource | Adds or updates a tag on a given resource |
| test_invoke_authorizer | Simulate the execution of an Authorizer in your RestApi with headers, parameters, and an in |
| test_invoke_method | Simulate the execution of a Method in your RestApi with headers, parameters, and an incom |
| untag_resource | Removes a tag from a given resource |
| update_account | Changes information about the current Account resource |
| update_api_key | Changes information about an ApiKey resource |
| update_authorizer | Updates an existing Authorizer resource |
| update_base_path_mapping | Changes information about the BasePathMapping resource |
| update_client_certificate | Changes information about an ClientCertificate resource |
| update_deployment | Changes information about a Deployment resource |
| update_documentation_part | Update documentation part |
| update_documentation_version | Update documentation version |
| update_domain_name | Changes information about the DomainName resource |
| update_gateway_response | Updates a GatewayResponse of a specified response type on the given RestApi |
| update_integration | Represents an update integration |
| update_integration_response | Represents an update integration response |
| update_method | Updates an existing Method resource |
| update_method_response | Updates an existing MethodResponse resource |
| update_model | Changes information about a model |
| update_request_validator | Updates a RequestValidator of a given RestApi |
| update_resource | Changes information about a Resource resource |
| update_rest_api | Changes information about the specified API |
| update_stage | Changes information about a Stage resource |
| update_usage | Grants a temporary extension to the remaining quota of a usage plan associated with a speci |
| update_usage_plan | Updates a usage plan of a given plan Id |
| update_vpc_link | Updates an existing VpcLink of a specified identifier |

## Examples

```
## Not run:
svc <- apigateway()
svc$create_api_key(
  Foo = 123
)

## End(Not run)
```

---

`apigatewaymanagementapi` *AmazonApiGatewayManagementApi*

---

## Description

The Amazon API Gateway Management API allows you to directly manage runtime aspects of your deployed APIs. To use it, you must explicitly set the SDK's endpoint to point to the endpoint of your deployed API. The endpoint will be of the form https://{api-id}.execute-api.{region}.amazonaws.com/{stage}, or will be the endpoint corresponding to your API's custom domain and base path, if applicable.

## Usage

```
apigatewaymanagementapi(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- apigatewaymanagementapi(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
```

```
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| delete_connection | Delete the connection with the provided id |
| get_connection | Get information about the connection with the provided id |
| post_to_connection | Sends the provided data to the specified connection |

## Examples

```
## Not run:
svc <- apigatewaymanagementapi()
svc$delete_connection(
  Foo = 123
)

## End(Not run)
```

---

`apigatewayv2` *AmazonApiGatewayV2*

---

## Description

Amazon API Gateway V2

## Usage

```
apigatewayv2(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- apigatewayv2(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| create_api | Creates an Api resource |
| create_api_mapping | Creates an API mapping |
| create_authorizer | Creates an Authorizer for an API |
| create_deployment | Creates a Deployment for an API |
| create_domain_name | Creates a domain name |
| create_integration | Creates an Integration |
| create_integration_response | Creates an IntegrationResponses |
| create_model | Creates a Model for an API |
| create_route | Creates a Route for an API |
| create_route_response | Creates a RouteResponse for a Route |
| create_stage | Creates a Stage for an API |
| create_vpc_link | Creates a VPC link |
| delete_access_log_settings | Deletes the AccessLogSettings for a Stage |
| delete_api | Deletes an Api resource |
| delete_api_mapping | Deletes an API mapping |
| delete_authorizer | Deletes an Authorizer |
| delete_cors_configuration | Deletes a CORS configuration |
| delete_deployment | Deletes a Deployment |
| delete_domain_name | Deletes a domain name |
| delete_integration | Deletes an Integration |
| delete_integration_response | Deletes an IntegrationResponses |
| delete_model | Deletes a Model |
| delete_route | Deletes a Route |
| delete_route_request_parameter | Deletes a route request parameter |
| delete_route_response | Deletes a RouteResponse |
| delete_route_settings | Deletes the RouteSettings for a stage |
| delete_stage | Deletes a Stage |
| delete_vpc_link | Deletes a VPC link |
| export_api | Export api |
| get_api | Gets an Api resource |

| | |
|---|---|
| get_api_mapping | Gets an API mapping |
| get_api_mappings | Gets API mappings |
| get_apis | Gets a collection of Api resources |
| get_authorizer | Gets an Authorizer |
| get_authorizers | Gets the Authorizers for an API |
| get_deployment | Gets a Deployment |
| get_deployments | Gets the Deployments for an API |
| get_domain_name | Gets a domain name |
| get_domain_names | Gets the domain names for an AWS account |
| get_integration | Gets an Integration |
| get_integration_response | Gets an IntegrationResponses |
| get_integration_responses | Gets the IntegrationResponses for an Integration |
| get_integrations | Gets the Integrations for an API |
| get_model | Gets a Model |
| get_models | Gets the Models for an API |
| get_model_template | Gets a model template |
| get_route | Gets a Route |
| get_route_response | Gets a RouteResponse |
| get_route_responses | Gets the RouteResponses for a Route |
| get_routes | Gets the Routes for an API |
| get_stage | Gets a Stage |
| get_stages | Gets the Stages for an API |
| get_tags | Gets a collection of Tag resources |
| get_vpc_link | Gets a VPC link |
| get_vpc_links | Gets a collection of VPC links |
| import_api | Imports an API |
| reimport_api | Puts an Api resource |
| reset_authorizers_cache | Resets all authorizer cache entries on a stage |
| tag_resource | Creates a new Tag resource to represent a tag |
| untag_resource | Deletes a Tag |
| update_api | Updates an Api resource |
| update_api_mapping | The API mapping |
| update_authorizer | Updates an Authorizer |
| update_deployment | Updates a Deployment |
| update_domain_name | Updates a domain name |
| update_integration | Updates an Integration |
| update_integration_response | Updates an IntegrationResponses |
| update_model | Updates a Model |
| update_route | Updates a Route |
| update_route_response | Updates a RouteResponse |
| update_stage | Updates a Stage |
| update_vpc_link | Updates a VPC link |

## Examples

```
## Not run:
svc <- apigatewayv2()
```

```
svc$create_api(
  Foo = 123
)

## End(Not run)
```

---

appmesh *AWS App Mesh*

---

## Description

AWS App Mesh is a service mesh based on the Envoy proxy that makes it easy to monitor and control microservices. App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high availability for your applications.

App Mesh gives you consistent visibility and network traffic controls for every microservice in an application. You can use App Mesh with AWS Fargate, Amazon ECS, Amazon EKS, Kubernetes on AWS, and Amazon EC2.

App Mesh supports microservice applications that use service discovery naming for their components. For more information about service discovery on Amazon ECS, see Service Discovery in the *Amazon Elastic Container Service Developer Guide*. Kubernetes `kube-dns` and `coredns` are supported. For more information, see DNS for Services and Pods in the Kubernetes documentation.

## Usage

```
appmesh(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- appmesh(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
```

```
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| create_gateway_route | Creates a gateway route |
| create_mesh | Creates a service mesh |
| create_route | Creates a route that is associated with a virtual router |
| create_virtual_gateway | Creates a virtual gateway |
| create_virtual_node | Creates a virtual node within a service mesh |
| create_virtual_router | Creates a virtual router within a service mesh |
| create_virtual_service | Creates a virtual service within a service mesh |
| delete_gateway_route | Deletes an existing gateway route |
| delete_mesh | Deletes an existing service mesh |
| delete_route | Deletes an existing route |
| delete_virtual_gateway | Deletes an existing virtual gateway |
| delete_virtual_node | Deletes an existing virtual node |
| delete_virtual_router | Deletes an existing virtual router |
| delete_virtual_service | Deletes an existing virtual service |
| describe_gateway_route | Describes an existing gateway route |
| describe_mesh | Describes an existing service mesh |
| describe_route | Describes an existing route |
| describe_virtual_gateway | Describes an existing virtual gateway |
| describe_virtual_node | Describes an existing virtual node |
| describe_virtual_router | Describes an existing virtual router |
| describe_virtual_service | Describes an existing virtual service |
| list_gateway_routes | Returns a list of existing gateway routes that are associated to a virtual gateway |
| list_meshes | Returns a list of existing service meshes |
| list_routes | Returns a list of existing routes in a service mesh |
| list_tags_for_resource | List the tags for an App Mesh resource |
| list_virtual_gateways | Returns a list of existing virtual gateways in a service mesh |
| list_virtual_nodes | Returns a list of existing virtual nodes |
| list_virtual_routers | Returns a list of existing virtual routers in a service mesh |
| list_virtual_services | Returns a list of existing virtual services in a service mesh |
| tag_resource | Associates the specified tags to a resource with the specified resourceArn |
| untag_resource | Deletes specified tags from a resource |
| update_gateway_route | Updates an existing gateway route that is associated to a specified virtual gateway in a service mes |
| update_mesh | Updates an existing service mesh |
| update_route | Updates an existing route for a specified service mesh and virtual router |
| update_virtual_gateway | Updates an existing virtual gateway in a specified service mesh |
| update_virtual_node | Updates an existing virtual node in a specified service mesh |
| update_virtual_router | Updates an existing virtual router in a specified service mesh |
| update_virtual_service | Updates an existing virtual service in a specified service mesh |

## Examples

```
## Not run:
svc <- appmesh()
svc$create_gateway_route(
  Foo = 123
)

## End(Not run)
```

---

`cloudfront` *Amazon CloudFront*

---

## Description

This is the *Amazon CloudFront API Reference*. This guide is for developers who need detailed information about CloudFront API actions, data types, and errors. For detailed information about CloudFront features, see the *Amazon CloudFront Developer Guide*.

## Usage

```
cloudfront(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- cloudfront(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
```

```
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| create_cache_policy | Creates a cache policy |
| create_cloud_front_origin_access_identity | Creates a new origin access identity |
| create_distribution | Creates a new web distribution |
| create_distribution_with_tags | Create a new distribution with tags |
| create_field_level_encryption_config | Create a new field-level encryption configuration |
| create_field_level_encryption_profile | Create a field-level encryption profile |
| create_invalidation | Create a new invalidation |
| create_key_group | Creates a key group that you can use with CloudFront signed URLs and sign |
| create_monitoring_subscription | Enables additional CloudWatch metrics for the specified CloudFront distribu |
| create_origin_request_policy | Creates an origin request policy |
| create_public_key | Uploads a public key to CloudFront that you can use with signed URLs and |
| create_realtime_log_config | Creates a real-time log configuration |
| create_streaming_distribution | This API is deprecated |
| create_streaming_distribution_with_tags | This API is deprecated |
| delete_cache_policy | Deletes a cache policy |
| delete_cloud_front_origin_access_identity | Delete an origin access identity |
| delete_distribution | Delete a distribution |
| delete_field_level_encryption_config | Remove a field-level encryption configuration |
| delete_field_level_encryption_profile | Remove a field-level encryption profile |
| delete_key_group | Deletes a key group |
| delete_monitoring_subscription | Disables additional CloudWatch metrics for the specified CloudFront distrib |
| delete_origin_request_policy | Deletes an origin request policy |
| delete_public_key | Remove a public key you previously added to CloudFront |
| delete_realtime_log_config | Deletes a real-time log configuration |
| delete_streaming_distribution | Delete a streaming distribution |
| get_cache_policy | Gets a cache policy, including the following metadata: |
| get_cache_policy_config | Gets a cache policy configuration |
| get_cloud_front_origin_access_identity | Get the information about an origin access identity |
| get_cloud_front_origin_access_identity_config | Get the configuration information about an origin access identity |
| get_distribution | Get the information about a distribution |
| get_distribution_config | Get the configuration information about a distribution |
| get_field_level_encryption | Get the field-level encryption configuration information |
| get_field_level_encryption_config | Get the field-level encryption configuration information |
| get_field_level_encryption_profile | Get the field-level encryption profile information |
| get_field_level_encryption_profile_config | Get the field-level encryption profile configuration information |
| get_invalidation | Get the information about an invalidation |
| get_key_group | Gets a key group, including the date and time when the key group was last n |
| get_key_group_config | Gets a key group configuration |
| get_monitoring_subscription | Gets information about whether additional CloudWatch metrics are enabled |
| get_origin_request_policy | Gets an origin request policy, including the following metadata: |
| get_origin_request_policy_config | Gets an origin request policy configuration |
| get_public_key | Gets a public key |

| | |
|---|---|
| get_public_key_config | Gets a public key configuration |
| get_realtime_log_config | Gets a real-time log configuration |
| get_streaming_distribution | Gets information about a specified RTMP distribution, including the distribu |
| get_streaming_distribution_config | Get the configuration information about a streaming distribution |
| list_cache_policies | Gets a list of cache policies |
| list_cloud_front_origin_access_identities | Lists origin access identities |
| list_distributions | List CloudFront distributions |
| list_distributions_by_cache_policy_id | Gets a list of distribution IDs for distributions that have a cache behavior tha |
| list_distributions_by_key_group | Gets a list of distribution IDs for distributions that have a cache behavior tha |
| list_distributions_by_origin_request_policy_id | Gets a list of distribution IDs for distributions that have a cache behavior tha |
| list_distributions_by_realtime_log_config | Gets a list of distributions that have a cache behavior that's associated with t |
| list_distributions_by_web_acl_id | List the distributions that are associated with a specified AWS WAF web AC |
| list_field_level_encryption_configs | List all field-level encryption configurations that have been created in Cloud |
| list_field_level_encryption_profiles | Request a list of field-level encryption profiles that have been created in Clo |
| list_invalidations | Lists invalidation batches |
| list_key_groups | Gets a list of key groups |
| list_origin_request_policies | Gets a list of origin request policies |
| list_public_keys | List all public keys that have been added to CloudFront for this account |
| list_realtime_log_configs | Gets a list of real-time log configurations |
| list_streaming_distributions | List streaming distributions |
| list_tags_for_resource | List tags for a CloudFront resource |
| tag_resource | Add tags to a CloudFront resource |
| untag_resource | Remove tags from a CloudFront resource |
| update_cache_policy | Updates a cache policy configuration |
| update_cloud_front_origin_access_identity | Update an origin access identity |
| update_distribution | Updates the configuration for a web distribution |
| update_field_level_encryption_config | Update a field-level encryption configuration |
| update_field_level_encryption_profile | Update a field-level encryption profile |
| update_key_group | Updates a key group |
| update_origin_request_policy | Updates an origin request policy configuration |
| update_public_key | Update public key information |
| update_realtime_log_config | Updates a real-time log configuration |
| update_streaming_distribution | Update a streaming distribution |

## Examples

```
## Not run:
svc <- cloudfront()
svc$create_cache_policy(
  Foo = 123
)

## End(Not run)
```

directconnect *AWS Direct Connect*

## Description

AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard Ethernet fiber-optic cable. One end of the cable is connected to your router, the other to an AWS Direct Connect router. With this connection in place, you can create virtual interfaces directly to the AWS cloud (for example, to Amazon EC2 and Amazon S3) and to Amazon VPC, bypassing Internet service providers in your network path. A connection provides access to all AWS Regions except the China (Beijing) and (China) Ningxia Regions. AWS resources in the China Regions can only be accessed through locations associated with those Regions.

## Usage

```
directconnect(config = list())
```

## Arguments

| | |
|---|---|
| config | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- directconnect(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| accept_direct_connect_gateway_association_proposal | Accepts a proposal request to attach a virtual private gateway or tr |
| allocate_connection_on_interconnect | Deprecated |
| allocate_hosted_connection | Creates a hosted connection on the specified interconnect or a link |
| allocate_private_virtual_interface | Provisions a private virtual interface to be owned by the specified |
| allocate_public_virtual_interface | Provisions a public virtual interface to be owned by the specified A |
| allocate_transit_virtual_interface | Provisions a transit virtual interface to be owned by the specified A |
| associate_connection_with_lag | Associates an existing connection with a link aggregation group (L |
| associate_hosted_connection | Associates a hosted connection and its virtual interfaces with a link |
| associate_virtual_interface | Associates a virtual interface with a specified link aggregation grou |
| confirm_connection | Confirms the creation of the specified hosted connection on an inte |
| confirm_private_virtual_interface | Accepts ownership of a private virtual interface created by another |
| confirm_public_virtual_interface | Accepts ownership of a public virtual interface created by another |
| confirm_transit_virtual_interface | Accepts ownership of a transit virtual interface created by another |
| create_bgp_peer | Creates a BGP peer on the specified virtual interface |
| create_connection | Creates a connection between a customer network and a specific A |
| create_direct_connect_gateway | Creates a Direct Connect gateway, which is an intermediate object |
| create_direct_connect_gateway_association | Creates an association between a Direct Connect gateway and a vi |
| create_direct_connect_gateway_association_proposal | Creates a proposal to associate the specified virtual private gatewa |
| create_interconnect | Creates an interconnect between an AWS Direct Connect Partner's |
| create_lag | Creates a link aggregation group (LAG) with the specified number |
| create_private_virtual_interface | Creates a private virtual interface |
| create_public_virtual_interface | Creates a public virtual interface |
| create_transit_virtual_interface | Creates a transit virtual interface |
| delete_bgp_peer | Deletes the specified BGP peer on the specified virtual interface w |
| delete_connection | Deletes the specified connection |
| delete_direct_connect_gateway | Deletes the specified Direct Connect gateway |
| delete_direct_connect_gateway_association | Deletes the association between the specified Direct Connect gatev |
| delete_direct_connect_gateway_association_proposal | Deletes the association proposal request between the specified Dir |
| delete_interconnect | Deletes the specified interconnect |
| delete_lag | Deletes the specified link aggregation group (LAG) |
| delete_virtual_interface | Deletes a virtual interface |
| describe_connection_loa | Deprecated |
| describe_connections | Displays the specified connection or all connections in this Region |
| describe_connections_on_interconnect | Deprecated |
| describe_direct_connect_gateway_association_proposals | Describes one or more association proposals for connection betwe |
| describe_direct_connect_gateway_associations | Lists the associations between your Direct Connect gateways and v |
| describe_direct_connect_gateway_attachments | Lists the attachments between your Direct Connect gateways and v |
| describe_direct_connect_gateways | Lists all your Direct Connect gateways or only the specified Direct |
| describe_hosted_connections | Lists the hosted connections that have been provisioned on the spe |
| describe_interconnect_loa | Deprecated |
| describe_interconnects | Lists the interconnects owned by the AWS account or only the spe |
| describe_lags | Describes all your link aggregation groups (LAG) or the specified |
| describe_loa | Gets the LOA-CFA for a connection, interconnect, or link aggrega |
| describe_locations | Lists the AWS Direct Connect locations in the current AWS Regio |
| describe_tags | Describes the tags associated with the specified AWS Direct Conn |
| describe_virtual_gateways | Lists the virtual private gateways owned by the AWS account |

| | |
|---|---|
| describe_virtual_interfaces | Displays all virtual interfaces for an AWS account |
| disassociate_connection_from_lag | Disassociates a connection from a link aggregation group (LAG) |
| list_virtual_interface_test_history | Lists the virtual interface failover test history |
| start_bgp_failover_test | Starts the virtual interface failover test that verifies your configurat |
| stop_bgp_failover_test | Stops the virtual interface failover test |
| tag_resource | Adds the specified tags to the specified AWS Direct Connect resou |
| untag_resource | Removes one or more tags from the specified AWS Direct Connec |
| update_direct_connect_gateway_association | Updates the specified attributes of the Direct Connect gateway ass |
| update_lag | Updates the attributes of the specified link aggregation group (LA |
| update_virtual_interface_attributes | Updates the specified attributes of the specified virtual private inte |

## Examples

```
## Not run:
svc <- directconnect()
svc$accept_direct_connect_gateway_association_proposal(
  Foo = 123
)

## End(Not run)
```

---

# elb *Elastic Load Balancing*

---

## Description

A load balancer can distribute incoming traffic across your EC2 instances. This enables you to increase the availability of your application. The load balancer also monitors the health of its registered instances and ensures that it routes traffic only to healthy instances. You configure your load balancer to accept incoming traffic by specifying one or more listeners, which are configured with a protocol and port number for connections from clients to the load balancer and a protocol and port number for connections from the load balancer to the instances.

Elastic Load Balancing supports three types of load balancers: Application Load Balancers, Network Load Balancers, and Classic Load Balancers. You can select a load balancer based on your application needs. For more information, see the Elastic Load Balancing User Guide.

This reference covers the 2012-06-01 API, which supports Classic Load Balancers. The 2015-12-01 API supports Application Load Balancers and Network Load Balancers.

To get started, create a load balancer with one or more listeners using `create_load_balancer`. Register your instances with the load balancer using `register_instances_with_load_balancer`.

All Elastic Load Balancing operations are *idempotent*, which means that they complete at most one time. If you repeat an operation, it succeeds with a 200 OK response code.

## Usage

```
elb(config = list())
```

## Arguments

| | |
|---|---|
| config | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- elb(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| add_tags | Adds the specified tags to the specified load balancer |
| apply_security_groups_to_load_balancer | Associates one or more security groups with your load balancer in a virtual |
| attach_load_balancer_to_subnets | Adds one or more subnets to the set of configured subnets for the specified |
| configure_health_check | Specifies the health check settings to use when evaluating the health state o |
| create_app_cookie_stickiness_policy | Generates a stickiness policy with sticky session lifetimes that follow that c |
| create_lb_cookie_stickiness_policy | Generates a stickiness policy with sticky session lifetimes controlled by the |
| create_load_balancer | Creates a Classic Load Balancer |
| create_load_balancer_listeners | Creates one or more listeners for the specified load balancer |
| create_load_balancer_policy | Creates a policy with the specified attributes for the specified load balancer |
| delete_load_balancer | Deletes the specified load balancer |
| delete_load_balancer_listeners | Deletes the specified listeners from the specified load balancer |
| delete_load_balancer_policy | Deletes the specified policy from the specified load balancer |
| deregister_instances_from_load_balancer | Deregisters the specified instances from the specified load balancer |
| describe_account_limits | Describes the current Elastic Load Balancing resource limits for your AWS |
| describe_instance_health | Describes the state of the specified instances with respect to the specified lo |
| describe_load_balancer_attributes | Describes the attributes for the specified load balancer |

| | |
|---|---|
| describe_load_balancer_policies | Describes the specified policies |
| describe_load_balancer_policy_types | Describes the specified load balancer policy types or all load balancer polic |
| describe_load_balancers | Describes the specified the load balancers |
| describe_tags | Describes the tags associated with the specified load balancers |
| detach_load_balancer_from_subnets | Removes the specified subnets from the set of configured subnets for the lo |
| disable_availability_zones_for_load_balancer | Removes the specified Availability Zones from the set of Availability Zones |
| enable_availability_zones_for_load_balancer | Adds the specified Availability Zones to the set of Availability Zones for th |
| modify_load_balancer_attributes | Modifies the attributes of the specified load balancer |
| register_instances_with_load_balancer | Adds the specified instances to the specified load balancer |
| remove_tags | Removes one or more tags from the specified load balancer |
| set_load_balancer_listener_ssl_certificate | Sets the certificate that terminates the specified listener's SSL connections |
| set_load_balancer_policies_for_backend_server | Replaces the set of policies associated with the specified port on which the |
| set_load_balancer_policies_of_listener | Replaces the current set of policies for the specified load balancer port with |

## Examples

```
## Not run:
svc <- elb()
# This example adds two tags to the specified load balancer.
svc$add_tags(
  LoadBalancerNames = list(
    "my-load-balancer"
  ),
  Tags = list(
    list(
      Key = "project",
      Value = "lima"
    ),
    list(
      Key = "department",
      Value = "digital-media"
    )
  )
)

## End(Not run)
```

---

elbv2 *Elastic Load Balancing*

---

## Description

A load balancer distributes incoming traffic across targets, such as your EC2 instances. This enables you to increase the availability of your application. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. You configure your

load balancer to accept incoming traffic by specifying one or more listeners, which are configured with a protocol and port number for connections from clients to the load balancer. You configure a target group with a protocol and port number for connections from the load balancer to the targets, and with health check settings to be used when checking the health status of the targets.

Elastic Load Balancing supports the following types of load balancers: Application Load Balancers, Network Load Balancers, Gateway Load Balancers, and Classic Load Balancers. This reference covers the following load balancer types:

- Application Load Balancer - Operates at the application layer (layer 7) and supports HTTP and HTTPS.
- Network Load Balancer - Operates at the transport layer (layer 4) and supports TCP, TLS, and UDP.
- Gateway Load Balancer - Operates at the network layer (layer 3).

For more information, see the Elastic Load Balancing User Guide.

All Elastic Load Balancing operations are idempotent, which means that they complete at most one time. If you repeat an operation, it succeeds.

## Usage

```
elbv2(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- elbv2(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| add_listener_certificates | Adds the specified SSL server certificate to the certificate list for the specified HTTPS or |
| add_tags | Adds the specified tags to the specified Elastic Load Balancing resource |
| create_listener | Creates a listener for the specified Application Load Balancer, Network Load Balancer |
| create_load_balancer | Creates an Application Load Balancer, Network Load Balancer, or Gateway Load Balanc |
| create_rule | Creates a rule for the specified listener |
| create_target_group | Creates a target group |
| delete_listener | Deletes the specified listener |
| delete_load_balancer | Deletes the specified Application Load Balancer, Network Load Balancer, or Gateway Lo |
| delete_rule | Deletes the specified rule |
| delete_target_group | Deletes the specified target group |
| deregister_targets | Deregisters the specified targets from the specified target group |
| describe_account_limits | Describes the current Elastic Load Balancing resource limits for your AWS account |
| describe_listener_certificates | Describes the default certificate and the certificate list for the specified HTTPS or TLS lis |
| describe_listeners | Describes the specified listeners or the listeners for the specified Application Load Balanc |
| describe_load_balancer_attributes | Describes the attributes for the specified Application Load Balancer, Network Load Balar |
| describe_load_balancers | Describes the specified load balancers or all of your load balancers |
| describe_rules | Describes the specified rules or the rules for the specified listener |
| describe_ssl_policies | Describes the specified policies or all policies used for SSL negotiation |
| describe_tags | Describes the tags for the specified Elastic Load Balancing resources |
| describe_target_group_attributes | Describes the attributes for the specified target group |
| describe_target_groups | Describes the specified target groups or all of your target groups |
| describe_target_health | Describes the health of the specified targets or all of your targets |
| modify_listener | Replaces the specified properties of the specified listener |
| modify_load_balancer_attributes | Modifies the specified attributes of the specified Application Load Balancer, Network Loa |
| modify_rule | Replaces the specified properties of the specified rule |
| modify_target_group | Modifies the health checks used when evaluating the health state of the targets in the spec |
| modify_target_group_attributes | Modifies the specified attributes of the specified target group |
| register_targets | Registers the specified targets with the specified target group |
| remove_listener_certificates | Removes the specified certificate from the certificate list for the specified HTTPS or TLS |
| remove_tags | Removes the specified tags from the specified Elastic Load Balancing resources |
| set_ip_address_type | Sets the type of IP addresses used by the subnets of the specified Application Load Balan |
| set_rule_priorities | Sets the priorities of the specified rules |
| set_security_groups | Associates the specified security groups with the specified Application Load Balancer |
| set_subnets | Enables the Availability Zones for the specified public subnets for the specified Applicati |

## Examples

```
## Not run:
svc <- elbv2()
# This example adds the specified tags to the specified load balancer.
svc$add_tags(
  ResourceArns = list(
    "arn:aws:elasticloadbalancing:us-west-2:123456789012:loadbalancer/app/m..."
  ),
  Tags = list(
```

```
    list(
      Key = "project",
      Value = "lima"
    ),
    list(
      Key = "department",
      Value = "digital-media"
    )
  )
)

## End(Not run)
```

---

`globalaccelerator` *AWS Global Accelerator*

---

## Description

This is the *AWS Global Accelerator API Reference*. This guide is for developers who need detailed information about AWS Global Accelerator API actions, data types, and errors. For more information about Global Accelerator features, see the AWS Global Accelerator Developer Guide.

AWS Global Accelerator is a service in which you create *accelerators* to improve the performance of your applications for local and global users. Depending on the type of accelerator you choose, you can gain additional benefits.

- By using a standard accelerator, you can improve availability of your internet applications that are used by a global audience. With a standard accelerator, Global Accelerator directs traffic to optimal endpoints over the AWS global network.
- For other scenarios, you might choose a custom routing accelerator. With a custom routing accelerator, you can use application logic to directly map one or more users to a specific endpoint among many endpoints.

Global Accelerator is a global service that supports endpoints in multiple AWS Regions but you must specify the US West (Oregon) Region to create or update accelerators.

By default, Global Accelerator provides you with two static IP addresses that you associate with your accelerator. With a standard accelerator, instead of using the IP addresses that Global Accelerator provides, you can configure these entry points to be IPv4 addresses from your own IP address ranges that you bring to Global Accelerator. The static IP addresses are anycast from the AWS edge network. For a standard accelerator, they distribute incoming application traffic across multiple endpoint resources in multiple AWS Regions, which increases the availability of your applications. Endpoints for standard accelerators can be Network Load Balancers, Application Load Balancers, Amazon EC2 instances, or Elastic IP addresses that are located in one AWS Region or multiple Regions. For custom routing accelerators, you map traffic that arrives to the static IP addresses to specific Amazon EC2 servers in endpoints that are virtual private cloud (VPC) subnets.

The static IP addresses remain assigned to your accelerator for as long as it exists, even if you disable the accelerator and it no longer accepts or routes traffic. However, when you *delete* an

accelerator, you lose the static IP addresses that are assigned to it, so you can no longer route traffic by using them. You can use IAM policies like tag-based permissions with Global Accelerator to limit the users who have permissions to delete an accelerator. For more information, see Tag-based policies.

For standard accelerators, Global Accelerator uses the AWS global network to route traffic to the optimal regional endpoint based on health, client location, and policies that you configure. The service reacts instantly to changes in health or configuration to ensure that internet traffic from clients is always directed to healthy endpoints.

For a list of the AWS Regions where Global Accelerator and other services are currently supported, see the AWS Region Table.

AWS Global Accelerator includes the following components:

**Static IP addresses:**

Global Accelerator provides you with a set of two static IP addresses that are anycast from the AWS edge network. If you bring your own IP address range to AWS (BYOIP) to use with a standard accelerator, you can instead assign IP addresses from your own pool to use with your accelerator. For more information, see Bring your own IP addresses (BYOIP) in AWS Global Accelerator.

The IP addresses serve as single fixed entry points for your clients. If you already have Elastic Load Balancing load balancers, Amazon EC2 instances, or Elastic IP address resources set up for your applications, you can easily add those to a standard accelerator in Global Accelerator. This allows Global Accelerator to use static IP addresses to access the resources.

The static IP addresses remain assigned to your accelerator for as long as it exists, even if you disable the accelerator and it no longer accepts or routes traffic. However, when you *delete* an accelerator, you lose the static IP addresses that are assigned to it, so you can no longer route traffic by using them. You can use IAM policies like tag-based permissions with Global Accelerator to delete an accelerator. For more information, see Tag-based policies.

**Accelerator:**

An accelerator directs traffic to endpoints over the AWS global network to improve the performance of your internet applications. Each accelerator includes one or more listeners.

There are two types of accelerators:

- A *standard* accelerator directs traffic to the optimal AWS endpoint based on several factors, including the user's location, the health of the endpoint, and the endpoint weights that you configure. This improves the availability and performance of your applications. Endpoints can be Network Load Balancers, Application Load Balancers, Amazon EC2 instances, or Elastic IP addresses.
- A *custom routing* accelerator directs traffic to one of possibly thousands of Amazon EC2 instances running in a single or multiple virtual private clouds (VPCs). With custom routing, listener ports are mapped to statically associate port ranges with VPC subnets, which allows Global Accelerator to determine an EC2 instance IP address at the time of connection. By default, all port mapping destinations in a VPC subnet can't receive traffic. You can choose to configure all destinations in the subnet to receive traffic, or to specify individual port mappings that can receive traffic.

For more information, see Types of accelerators.

**DNS name:**

Global Accelerator assigns each accelerator a default Domain Name System (DNS) name, similar to `a1234567890abcdef.awsglobalaccelerator.com`, that points to the static IP addresses that Global Accelerator assigns to you or that you choose from your own IP address range. Depending on the use case, you can use your accelerator's static IP addresses or DNS name to route traffic to your accelerator, or set up DNS records to route traffic using your own custom domain name.

**Network zone:**

A network zone services the static IP addresses for your accelerator from a unique IP subnet. Similar to an AWS Availability Zone, a network zone is an isolated unit with its own set of physical infrastructure. When you configure an accelerator, by default, Global Accelerator allocates two IPv4 addresses for it. If one IP address from a network zone becomes unavailable due to IP address blocking by certain client networks, or network disruptions, then client applications can retry on the healthy static IP address from the other isolated network zone.

**Listener:**

A listener processes inbound connections from clients to Global Accelerator, based on the port (or port range) and protocol (or protocols) that you configure. A listener can be configured for TCP, UDP, or both TCP and UDP protocols. Each listener has one or more endpoint groups associated with it, and traffic is forwarded to endpoints in one of the groups. You associate endpoint groups with listeners by specifying the Regions that you want to distribute traffic to. With a standard accelerator, traffic is distributed to optimal endpoints within the endpoint groups associated with a listener.

**Endpoint group:**

Each endpoint group is associated with a specific AWS Region. Endpoint groups include one or more endpoints in the Region. With a standard accelerator, you can increase or reduce the percentage of traffic that would be otherwise directed to an endpoint group by adjusting a setting called a *traffic dial*. The traffic dial lets you easily do performance testing or blue/green deployment testing, for example, for new releases across different AWS Regions.

**Endpoint:**

An endpoint is a resource that Global Accelerator directs traffic to.

Endpoints for standard accelerators can be Network Load Balancers, Application Load Balancers, Amazon EC2 instances, or Elastic IP addresses. An Application Load Balancer endpoint can be internet-facing or internal. Traffic for standard accelerators is routed to endpoints based on the health of the endpoint along with configuration options that you choose, such as endpoint weights. For each endpoint, you can configure weights, which are numbers that you can use to specify the proportion of traffic to route to each one. This can be useful, for example, to do performance testing within a Region.

Endpoints for custom routing accelerators are virtual private cloud (VPC) subnets with one or many EC2 instances.

## Usage

```
globalaccelerator(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- globalaccelerator(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| add_custom_routing_endpoints | Associate a virtual private cloud (VPC) subnet endpoint with your cust |
| advertise_byoip_cidr | Advertises an IPv4 address range that is provisioned for use with your |
| allow_custom_routing_traffic | Specify the Amazon EC2 instance (destination) IP addresses and ports |
| create_accelerator | Create an accelerator |
| create_custom_routing_accelerator | Create a custom routing accelerator |
| create_custom_routing_endpoint_group | Create an endpoint group for the specified listener for a custom routing |
| create_custom_routing_listener | Create a listener to process inbound connections from clients to a custo |
| create_endpoint_group | Create an endpoint group for the specified listener |
| create_listener | Create a listener to process inbound connections from clients to an acce |
| delete_accelerator | Delete an accelerator |
| delete_custom_routing_accelerator | Delete a custom routing accelerator |
| delete_custom_routing_endpoint_group | Delete an endpoint group from a listener for a custom routing accelerat |
| delete_custom_routing_listener | Delete a listener for a custom routing accelerator |
| delete_endpoint_group | Delete an endpoint group from a listener |
| delete_listener | Delete a listener from an accelerator |
| deny_custom_routing_traffic | Specify the Amazon EC2 instance (destination) IP addresses and ports |
| deprovision_byoip_cidr | Releases the specified address range that you provisioned to use with y |
| describe_accelerator | Describe an accelerator |
| describe_accelerator_attributes | Describe the attributes of an accelerator |
| describe_custom_routing_accelerator | Describe a custom routing accelerator |
| describe_custom_routing_accelerator_attributes | Describe the attributes of a custom routing accelerator |
| describe_custom_routing_endpoint_group | Describe an endpoint group for a custom routing accelerator |
| describe_custom_routing_listener | The description of a listener for a custom routing accelerator |
| describe_endpoint_group | Describe an endpoint group |

| | |
|---|---|
| describe_listener | Describe a listener |
| list_accelerators | List the accelerators for an AWS account |
| list_byoip_cidrs | Lists the IP address ranges that were specified in calls to ProvisionByoi |
| list_custom_routing_accelerators | List the custom routing accelerators for an AWS account |
| list_custom_routing_endpoint_groups | List the endpoint groups that are associated with a listener for a custom |
| list_custom_routing_listeners | List the listeners for a custom routing accelerator |
| list_custom_routing_port_mappings | Provides a complete mapping from the public accelerator IP address an |
| list_custom_routing_port_mappings_by_destination | List the port mappings for a specific EC2 instance (destination) in a VP |
| list_endpoint_groups | List the endpoint groups that are associated with a listener |
| list_listeners | List the listeners for an accelerator |
| list_tags_for_resource | List all tags for an accelerator |
| provision_byoip_cidr | Provisions an IP address range to use with your AWS resources through |
| remove_custom_routing_endpoints | Remove endpoints from a custom routing accelerator |
| tag_resource | Add tags to an accelerator resource |
| untag_resource | Remove tags from a Global Accelerator resource |
| update_accelerator | Update an accelerator |
| update_accelerator_attributes | Update the attributes for an accelerator |
| update_custom_routing_accelerator | Update a custom routing accelerator |
| update_custom_routing_accelerator_attributes | Update the attributes for a custom routing accelerator |
| update_custom_routing_listener | Update a listener for a custom routing accelerator |
| update_endpoint_group | Update an endpoint group |
| update_listener | Update a listener |
| withdraw_byoip_cidr | Stops advertising an address range that is provisioned as an address poo |

## Examples

```
## Not run:
svc <- globalaccelerator()
svc$add_custom_routing_endpoints(
  Foo = 123
)

## End(Not run)
```

---

`route53` *Amazon Route 53*

---

## Description

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service.

## Usage

```
route53(config = list())
```

## Arguments

| | |
|---|---|
| config | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- route53(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| activate_key_signing_key | Activates a key signing key (KSK) so that it can be used for signing by DNSS |
| associate_vpc_with_hosted_zone | Associates an Amazon VPC with a private hosted zone |
| change_resource_record_sets | Creates, changes, or deletes a resource record set, which contains authoritative |
| change_tags_for_resource | Adds, edits, or deletes tags for a health check or a hosted zone |
| create_health_check | Creates a new health check |
| create_hosted_zone | Creates a new public or private hosted zone |
| create_key_signing_key | Creates a new key signing key (KSK) associated with a hosted zone |
| create_query_logging_config | Creates a configuration for DNS query logging |
| create_reusable_delegation_set | Creates a delegation set (a group of four name servers) that can be reused by r |
| create_traffic_policy | Creates a traffic policy, which you use to create multiple DNS resource record |
| create_traffic_policy_instance | Creates resource record sets in a specified hosted zone based on the settings in |
| create_traffic_policy_version | Creates a new version of an existing traffic policy |
| create_vpc_association_authorization | Authorizes the AWS account that created a specified VPC to submit an Assoc |
| deactivate_key_signing_key | Deactivates a key signing key (KSK) so that it will not be used for signing by |
| delete_health_check | Deletes a health check |
| delete_hosted_zone | Deletes a hosted zone |
| delete_key_signing_key | Deletes a key signing key (KSK) |
| delete_query_logging_config | Deletes a configuration for DNS query logging |
| delete_reusable_delegation_set | Deletes a reusable delegation set |
| delete_traffic_policy | Deletes a traffic policy |

| | |
|---|---|
| delete_traffic_policy_instance | Deletes a traffic policy instance and all of the resource record sets that Amazo |
| delete_vpc_association_authorization | Removes authorization to submit an AssociateVPCWithHostedZone request t |
| disable_hosted_zone_dnssec | Disables DNSSEC signing in a specific hosted zone |
| disassociate_vpc_from_hosted_zone | Disassociates an Amazon Virtual Private Cloud (Amazon VPC) from an Ama |
| enable_hosted_zone_dnssec | Enables DNSSEC signing in a specific hosted zone |
| get_account_limit | Gets the specified limit for the current account, for example, the maximum nu |
| get_change | Returns the current status of a change batch request |
| get_checker_ip_ranges | GetCheckerIpRanges still works, but we recommend that you download ip-ra |
| get_dnssec | Returns information about DNSSEC for a specific hosted zone, including the |
| get_geo_location | Gets information about whether a specified geographic location is supported f |
| get_health_check | Gets information about a specified health check |
| get_health_check_count | Retrieves the number of health checks that are associated with the current AW |
| get_health_check_last_failure_reason | Gets the reason that a specified health check failed most recently |
| get_health_check_status | Gets status of a specified health check |
| get_hosted_zone | Gets information about a specified hosted zone including the four name serve |
| get_hosted_zone_count | Retrieves the number of hosted zones that are associated with the current AW |
| get_hosted_zone_limit | Gets the specified limit for a specified hosted zone, for example, the maximun |
| get_query_logging_config | Gets information about a specified configuration for DNS query logging |
| get_reusable_delegation_set | Retrieves information about a specified reusable delegation set, including the |
| get_reusable_delegation_set_limit | Gets the maximum number of hosted zones that you can associate with the sp |
| get_traffic_policy | Gets information about a specific traffic policy version |
| get_traffic_policy_instance | Gets information about a specified traffic policy instance |
| get_traffic_policy_instance_count | Gets the number of traffic policy instances that are associated with the current |
| list_geo_locations | Retrieves a list of supported geographic locations |
| list_health_checks | Retrieve a list of the health checks that are associated with the current AWS a |
| list_hosted_zones | Retrieves a list of the public and private hosted zones that are associated with |
| list_hosted_zones_by_name | Retrieves a list of your hosted zones in lexicographic order |
| list_hosted_zones_by_vpc | Lists all the private hosted zones that a specified VPC is associated with, rega |
| list_query_logging_configs | Lists the configurations for DNS query logging that are associated with the cu |
| list_resource_record_sets | Lists the resource record sets in a specified hosted zone |
| list_reusable_delegation_sets | Retrieves a list of the reusable delegation sets that are associated with the curr |
| list_tags_for_resource | Lists tags for one health check or hosted zone |
| list_tags_for_resources | Lists tags for up to 10 health checks or hosted zones |
| list_traffic_policies | Gets information about the latest version for every traffic policy that is associa |
| list_traffic_policy_instances | Gets information about the traffic policy instances that you created by using th |
| list_traffic_policy_instances_by_hosted_zone | Gets information about the traffic policy instances that you created in a specifi |
| list_traffic_policy_instances_by_policy | Gets information about the traffic policy instances that you created by using a |
| list_traffic_policy_versions | Gets information about all of the versions for a specified traffic policy |
| list_vpc_association_authorizations | Gets a list of the VPCs that were created by other accounts and that can be ass |
| test_dns_answer | Gets the value that Amazon Route 53 returns in response to a DNS request for |
| update_health_check | Updates an existing health check |
| update_hosted_zone_comment | Updates the comment for a specified hosted zone |
| update_traffic_policy_comment | Updates the comment for a specified traffic policy version |
| update_traffic_policy_instance | Updates the resource record sets in a specified hosted zone that were created b |

## Examples

```
## Not run:
svc <- route53()
# The following example associates the VPC with ID vpc-1a2b3c4d with the
# hosted zone with ID Z3M3LMPEXAMPLE.
svc$associate_vpc_with_hosted_zone(
  Comment = "",
  HostedZoneId = "Z3M3LMPEXAMPLE",
  VPC = list(
    VPCId = "vpc-1a2b3c4d",
    VPCRegion = "us-east-2"
  )
)

## End(Not run)
```

---

# route53domains *Amazon Route 53 Domains*

---

## Description

Amazon Route 53 API actions let you register domain names and perform related operations.

## Usage

```
route53domains(config = list())
```

## Arguments

| | |
|---|---|
| config | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- route53domains(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
```

```
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| accept_domain_transfer_from_another_aws_account | Accepts the transfer of a domain from another AWS account to the cur |
| cancel_domain_transfer_to_another_aws_account | Cancels the transfer of a domain from the current AWS account to anot |
| check_domain_availability | This operation checks the availability of one domain name |
| check_domain_transferability | Checks whether a domain name can be transferred to Amazon Route 5 |
| delete_tags_for_domain | This operation deletes the specified tags for a domain |
| disable_domain_auto_renew | This operation disables automatic renewal of domain registration for th |
| disable_domain_transfer_lock | This operation removes the transfer lock on the domain (specifically th |
| enable_domain_auto_renew | This operation configures Amazon Route 53 to automatically renew the |
| enable_domain_transfer_lock | This operation sets the transfer lock on the domain (specifically the clie |
| get_contact_reachability_status | For operations that require confirmation that the email address for the r |
| get_domain_detail | This operation returns detailed information about a specified domain th |
| get_domain_suggestions | The GetDomainSuggestions operation returns a list of suggested domai |
| get_operation_detail | This operation returns the current status of an operation that is not com |
| list_domains | This operation returns all the domain names registered with Amazon R |
| list_operations | Returns information about all of the operations that return an operation |
| list_tags_for_domain | This operation returns all of the tags that are associated with the specifi |
| register_domain | This operation registers a domain |
| reject_domain_transfer_from_another_aws_account | Rejects the transfer of a domain from another AWS account to the curr |
| renew_domain | This operation renews a domain for the specified number of years |
| resend_contact_reachability_email | For operations that require confirmation that the email address for the r |
| retrieve_domain_auth_code | This operation returns the AuthCode for the domain |
| transfer_domain | Transfers a domain from another registrar to Amazon Route 53 |
| transfer_domain_to_another_aws_account | Transfers a domain from the current AWS account to another AWS acc |
| update_domain_contact | This operation updates the contact information for a particular domain |
| update_domain_contact_privacy | This operation updates the specified domain contact's privacy setting |
| update_domain_nameservers | This operation replaces the current set of name servers for the domain |
| update_tags_for_domain | This operation adds or updates tags for a specified domain |
| view_billing | Returns all the domain-related billing records for the current AWS acc |

## Examples

```
## Not run:
svc <- route53domains()
svc$accept_domain_transfer_from_another_aws_account(
  Foo = 123
)

## End(Not run)
```

route53resolver *Amazon Route 53 Resolver*

## Description

When you create a VPC using Amazon VPC, you automatically get DNS resolution within the VPC from Route 53 Resolver. By default, Resolver answers DNS queries for VPC domain names such as domain names for EC2 instances or ELB load balancers. Resolver performs recursive lookups against public name servers for all other domain names.

You can also configure DNS resolution between your VPC and your network over a Direct Connect or VPN connection:

**Forward DNS queries from resolvers on your network to Route 53 Resolver**

DNS resolvers on your network can forward DNS queries to Resolver in a specified VPC. This allows your DNS resolvers to easily resolve domain names for AWS resources such as EC2 instances or records in a Route 53 private hosted zone. For more information, see How DNS Resolvers on Your Network Forward DNS Queries to Route 53 Resolver in the *Amazon Route 53 Developer Guide*.

**Conditionally forward queries from a VPC to resolvers on your network**

You can configure Resolver to forward queries that it receives from EC2 instances in your VPCs to DNS resolvers on your network. To forward selected queries, you create Resolver rules that specify the domain names for the DNS queries that you want to forward (such as example.com), and the IP addresses of the DNS resolvers on your network that you want to forward the queries to. If a query matches multiple rules (example.com, acme.example.com), Resolver chooses the rule with the most specific match (acme.example.com) and forwards the query to the IP addresses that you specified in that rule. For more information, see How Route 53 Resolver Forwards DNS Queries from Your VPCs to Your Network in the *Amazon Route 53 Developer Guide*.

Like Amazon VPC, Resolver is regional. In each region where you have VPCs, you can choose whether to forward queries from your VPCs to your network (outbound queries), from your network to your VPCs (inbound queries), or both.

## Usage

```
route53resolver(config = list())
```

## Arguments

| | |
|---|---|
| config | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

### Service syntax

```
svc <- route53resolver(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

### Operations

| | |
|---|---|
| associate_resolver_endpoint_ip_address | Adds IP addresses to an inbound or an outbound Resolver endpoint |
| associate_resolver_query_log_config | Associates an Amazon VPC with a specified query logging configuration |
| associate_resolver_rule | Associates a Resolver rule with a VPC |
| create_resolver_endpoint | Creates a Resolver endpoint |
| create_resolver_query_log_config | Creates a Resolver query logging configuration, which defines where you want |
| create_resolver_rule | For DNS queries that originate in your VPCs, specifies which Resolver endpoi |
| delete_resolver_endpoint | Deletes a Resolver endpoint |
| delete_resolver_query_log_config | Deletes a query logging configuration |
| delete_resolver_rule | Deletes a Resolver rule |
| disassociate_resolver_endpoint_ip_address | Removes IP addresses from an inbound or an outbound Resolver endpoint |
| disassociate_resolver_query_log_config | Disassociates a VPC from a query logging configuration |
| disassociate_resolver_rule | Removes the association between a specified Resolver rule and a specified VPC |
| get_resolver_dnssec_config | Gets DNSSEC validation information for a specified resource |
| get_resolver_endpoint | Gets information about a specified Resolver endpoint, such as whether it's an i |
| get_resolver_query_log_config | Gets information about a specified Resolver query logging configuration, such |
| get_resolver_query_log_config_association | Gets information about a specified association between a Resolver query loggi |
| get_resolver_query_log_config_policy | Gets information about a query logging policy |
| get_resolver_rule | Gets information about a specified Resolver rule, such as the domain name that |
| get_resolver_rule_association | Gets information about an association between a specified Resolver rule and a |
| get_resolver_rule_policy | Gets information about the Resolver rule policy for a specified rule |
| list_resolver_dnssec_configs | Lists the configurations for DNSSEC validation that are associated with the cu |
| list_resolver_endpoint_ip_addresses | Gets the IP addresses for a specified Resolver endpoint |
| list_resolver_endpoints | Lists all the Resolver endpoints that were created using the current AWS accou |
| list_resolver_query_log_config_associations | Lists information about associations between Amazon VPCs and query logging |
| list_resolver_query_log_configs | Lists information about the specified query logging configurations |
| list_resolver_rule_associations | Lists the associations that were created between Resolver rules and VPCs using |
| list_resolver_rules | Lists the Resolver rules that were created using the current AWS account |
| list_tags_for_resource | Lists the tags that you associated with the specified resource |
| put_resolver_query_log_config_policy | Specifies an AWS account that you want to share a query logging configuration |
| put_resolver_rule_policy | Specifies an AWS rule that you want to share with another account, the accoun |

| | |
|---|---|
| tag_resource | Adds one or more tags to a specified resource |
| untag_resource | Removes one or more tags from a specified resource |
| update_resolver_dnssec_config | Updates an existing DNSSEC validation configuration |
| update_resolver_endpoint | Updates the name of an inbound or an outbound Resolver endpoint |
| update_resolver_rule | Updates settings for a specified Resolver rule |

## Examples

```
## Not run:
svc <- route53resolver()
svc$associate_resolver_endpoint_ip_address(
  Foo = 123
)

## End(Not run)
```

---

`servicediscovery` *AWS Cloud Map*

---

## Description

AWS Cloud Map lets you configure public DNS, private DNS, or HTTP namespaces that your microservice applications run in. When an instance of the service becomes available, you can call the AWS Cloud Map API to register the instance with AWS Cloud Map. For public or private DNS namespaces, AWS Cloud Map automatically creates DNS records and an optional health check. Clients that submit public or private DNS queries, or HTTP requests, for the service receive an answer that contains up to eight healthy records.

## Usage

```
servicediscovery(config = list())
```

## Arguments

| | |
|---|---|
| `config` | Optional configuration of credentials, endpoint, and/or region. |

## Value

A client for the service. You can call the service's operations using syntax like `svc$operation(...)`, where `svc` is the name you've assigned to the client. The available operations are listed in the Operations section.

## Service syntax

```
svc <- servicediscovery(
  config = list(
    credentials = list(
      creds = list(
        access_key_id = "string",
        secret_access_key = "string",
        session_token = "string"
      ),
      profile = "string"
    ),
    endpoint = "string",
    region = "string"
  )
)
```

## Operations

| | |
|---|---|
| create_http_namespace | Creates an HTTP namespace |
| create_private_dns_namespace | Creates a private namespace based on DNS, which will be visible only inside a speci |
| create_public_dns_namespace | Creates a public namespace based on DNS, which will be visible on the internet |
| create_service | Creates a service, which defines the configuration for the following entities: |
| delete_namespace | Deletes a namespace from the current account |
| delete_service | Deletes a specified service |
| deregister_instance | Deletes the Amazon Route 53 DNS records and health check, if any, that AWS Clou |
| discover_instances | Discovers registered instances for a specified namespace and service |
| get_instance | Gets information about a specified instance |
| get_instances_health_status | Gets the current health status (Healthy, Unhealthy, or Unknown) of one or more insta |
| get_namespace | Gets information about a namespace |
| get_operation | Gets information about any operation that returns an operation ID in the response, su |
| get_service | Gets the settings for a specified service |
| list_instances | Lists summary information about the instances that you registered by using a specifi |
| list_namespaces | Lists summary information about the namespaces that were created by the current A |
| list_operations | Lists operations that match the criteria that you specify |
| list_services | Lists summary information for all the services that are associated with one or more s |
| list_tags_for_resource | Lists tags for the specified resource |
| register_instance | Creates or updates one or more records and, optionally, creates a health check based |
| tag_resource | Adds one or more tags to the specified resource |
| untag_resource | Removes one or more tags from the specified resource |
| update_instance_custom_health_status | Submits a request to change the health status of a custom health check to healthy or |
| update_service | Submits a request to perform the following operations: |

## Examples

```
## Not run:
svc <- servicediscovery()
```

```
# This example creates an HTTP namespace.
svc$create_http_namespace(
  CreatorRequestId = "example-creator-request-id-0001",
  Description = "Example.com AWS Cloud Map HTTP Namespace",
  Name = "example-http.com"
)

## End(Not run)
```

# Index