

Package ‘paws.application.integration’

August 23, 2021

Title 'Amazon Web Services' Application Integration Services

Version 0.1.12

Description Interface to 'Amazon Web Services' application integration services, including 'Simple Queue Service' ('SQS') message queue, 'Simple Notification Service' ('SNS') publish/subscribe messaging, 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 'eventbridge_service.R' 'eventbridge_interfaces.R'
'eventbridge_operations.R' 'mq_service.R' 'mq_interfaces.R'
'mq_operations.R' 'sfn_service.R' 'sfn_interfaces.R'
'sfn_operations.R' 'sns_service.R' 'sns_interfaces.R'
'sns_operations.R' 'sqs_service.R' 'sqs_interfaces.R'
'sqs_operations.R' 'swf_service.R' 'swf_interfaces.R'
'swf_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 07:11:15 UTC

R topics documented:

eventbridge	2
mq	4
sfn	6
sns	8
sqs	10
swf	12
Index	15

eventbridge	<i>Amazon EventBridge</i>
-------------	---------------------------

Description

Amazon EventBridge helps you to respond to state changes in your AWS resources. When your resources change state, they automatically send events into an event stream. You can create rules that match selected events in the stream and route them to targets to take action. You can also use rules to take action on a predetermined schedule. For example, you can configure rules to:

- Automatically invoke an AWS Lambda function to update DNS entries when an event notifies you that Amazon EC2 instance enters the running state.
- Direct specific API records from AWS CloudTrail to an Amazon Kinesis data stream for detailed analysis of potential security or availability risks.
- Periodically invoke a built-in target to create a snapshot of an Amazon EBS volume.

For more information about the features of Amazon EventBridge, see the [Amazon EventBridge User Guide](#).

Usage

```
eventbridge(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 <- eventbridge(
  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_event_source	Activates a partner event source that has been deactivated
cancel_replay	Cancels the specified replay
create_archive	Creates an archive of events with the specified settings
create_event_bus	Creates a new event bus within your account
create_partner_event_source	Called by an SaaS partner to create a partner event source
deactivate_event_source	You can use this operation to temporarily stop receiving events from the specified partner event source
delete_archive	Deletes the specified archive
delete_event_bus	Deletes the specified custom event bus or partner event bus
delete_partner_event_source	This operation is used by SaaS partners to delete a partner event source
delete_rule	Deletes the specified rule
describe_archive	Retrieves details about an archive
describe_event_bus	Displays details about an event bus in your account
describe_event_source	This operation lists details about a partner event source that is shared with your account
describe_partner_event_source	An SaaS partner can use this operation to list details about a partner event source that the partner has shared with your account
describe_replay	Retrieves details about a replay
describe_rule	Describes the specified rule
disable_rule	Disables the specified rule
enable_rule	Enables the specified rule
list_archives	Lists your archives
list_event_buses	Lists all the event buses in your account, including the default event bus, custom event bus, and partner event bus
list_event_sources	You can use this to see all the partner event sources that have been shared with your AWS account
list_partner_event_source_accounts	An SaaS partner can use this operation to display the AWS account ID that a particular partner event source is shared with
list_partner_event_sources	An SaaS partner can use this operation to list all the partner event source names that the partner has shared with your account
list_replays	Lists your replays
list_rule_names_by_target	Lists the rules for the specified target
list_rules	Lists your Amazon EventBridge rules
list_tags_for_resource	Displays the tags associated with an EventBridge resource
list_targets_by_rule	Lists the targets assigned to the specified rule
put_events	Sends custom events to Amazon EventBridge so that they can be matched to rules
put_partner_events	This is used by SaaS partners to write events to a customer's partner event bus

put_permission	Running PutPermission permits the specified AWS account or AWS organization to put
put_rule	Creates or updates the specified rule
put_targets	Adds the specified targets to the specified rule, or updates the targets if they are already
remove_permission	Revokes the permission of another AWS account to be able to put events to the specified
remove_targets	Removes the specified targets from the specified rule
start_replay	Starts the specified replay
tag_resource	Assigns one or more tags (key-value pairs) to the specified EventBridge resource
test_event_pattern	Tests whether the specified event pattern matches the provided event
untag_resource	Removes one or more tags from the specified EventBridge resource
update_archive	Updates the specified archive

Examples

```
## Not run:
svc <- eventbridge()
svc$activate_event_source(
  Foo = 123
)

## End(Not run)
```

mq

AmazonMQ

Description

Amazon MQ is a managed message broker service for Apache ActiveMQ and RabbitMQ that makes it easy to set up and operate message brokers in the cloud. A message broker allows software applications and components to communicate using various programming languages, operating systems, and formal messaging protocols.

Usage

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

```

Operations

<code>create_broker</code>	Creates a broker
<code>create_configuration</code>	Creates a new configuration for the specified configuration name
<code>create_tags</code>	Add a tag to a resource
<code>create_user</code>	Creates an ActiveMQ user
<code>delete_broker</code>	Deletes a broker
<code>delete_tags</code>	Removes a tag from a resource
<code>delete_user</code>	Deletes an ActiveMQ user
<code>describe_broker</code>	Returns information about the specified broker
<code>describe_broker_engine_types</code>	Describe available engine types and versions
<code>describe_broker_instance_options</code>	Describe available broker instance options
<code>describe_configuration</code>	Returns information about the specified configuration
<code>describe_configuration_revision</code>	Returns the specified configuration revision for the specified configuration
<code>describe_user</code>	Returns information about an ActiveMQ user
<code>list_brokers</code>	Returns a list of all brokers
<code>list_configuration_revisions</code>	Returns a list of all revisions for the specified configuration
<code>list_configurations</code>	Returns a list of all configurations
<code>list_tags</code>	Lists tags for a resource
<code>list_users</code>	Returns a list of all ActiveMQ users
<code>reboot_broker</code>	Reboots a broker
<code>update_broker</code>	Adds a pending configuration change to a broker
<code>update_configuration</code>	Updates the specified configuration
<code>update_user</code>	Updates the information for an ActiveMQ user

Examples

```

## Not run:
svc <- mq()
svc$create_broker(

```

```
    Foo = 123
  )

## End(Not run)
```

sfn

AWS Step Functions

Description

AWS Step Functions is a service that lets you coordinate the components of distributed applications and microservices using visual workflows.

You can use Step Functions to build applications from individual components, each of which performs a discrete function, or *task*, allowing you to scale and change applications quickly. Step Functions provides a console that helps visualize the components of your application as a series of steps. Step Functions automatically triggers and tracks each step, and retries steps when there are errors, so your application executes predictably and in the right order every time. Step Functions logs the state of each step, so you can quickly diagnose and debug any issues.

Step Functions manages operations and underlying infrastructure to ensure your application is available at any scale. You can run tasks on AWS, your own servers, or any system that has access to AWS. You can access and use Step Functions using the console, the AWS SDKs, or an HTTP API. For more information about Step Functions, see the [AWS Step Functions Developer Guide](#).

Usage

```
sfn(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 <- sfn(
  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_activity	Creates an activity
create_state_machine	Creates a state machine
delete_activity	Deletes an activity
delete_state_machine	Deletes a state machine
describe_activity	Describes an activity
describe_execution	Describes an execution
describe_state_machine	Describes a state machine
describe_state_machine_for_execution	Describes the state machine associated with a specific execution
get_activity_task	Used by workers to retrieve a task (with the specified activity ARN) which has been
get_execution_history	Returns the history of the specified execution as a list of events
list_activities	Lists the existing activities
list_executions	Lists the executions of a state machine that meet the filtering criteria
list_state_machines	Lists the existing state machines
list_tags_for_resource	List tags for a given resource
send_task_failure	Used by activity workers and task states using the callback pattern to report that the
send_task_heartbeat	Used by activity workers and task states using the callback pattern to report to Step F
send_task_success	Used by activity workers and task states using the callback pattern to report that the
start_execution	Starts a state machine execution
start_sync_execution	Starts a Synchronous Express state machine execution
stop_execution	Stops an execution
tag_resource	Add a tag to a Step Functions resource
untag_resource	Remove a tag from a Step Functions resource
update_state_machine	Updates an existing state machine by modifying its definition, roleArn, or loggingCo

Examples

```

## Not run:
svc <- sfn()
svc$create_activity(
  Foo = 123
)

## End(Not run)

```

Description

Amazon Simple Notification Service (Amazon SNS) is a web service that enables you to build distributed web-enabled applications. Applications can use Amazon SNS to easily push real-time notification messages to interested subscribers over multiple delivery protocols. For more information about this product see <https://aws.amazon.com/sns>. For detailed information about Amazon SNS features and their associated API calls, see the [Amazon SNS Developer Guide](#).

For information on the permissions you need to use this API, see [Identity and access management in Amazon SNS](#) in the *Amazon SNS Developer Guide*.

We also provide SDKs that enable you to access Amazon SNS from your preferred programming language. The SDKs contain functionality that automatically takes care of tasks such as: cryptographically signing your service requests, retrying requests, and handling error responses. For a list of available SDKs, go to [Tools for Amazon Web Services](#).

Usage

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


Operations

<code>add_permission</code>	Adds a statement to a topic's access control policy, granting access for the specified
<code>check_if_phone_number_is_opted_out</code>	Accepts a phone number and indicates whether the phone holder has opted out of re
<code>confirm_subscription</code>	Verifies an endpoint owner's intent to receive messages by validating the token sent
<code>create_platform_application</code>	Creates a platform application object for one of the supported push notification serv
<code>create_platform_endpoint</code>	Creates an endpoint for a device and mobile app on one of the supported push notif
<code>create_topic</code>	Creates a topic to which notifications can be published
<code>delete_endpoint</code>	Deletes the endpoint for a device and mobile app from Amazon SNS
<code>delete_platform_application</code>	Deletes a platform application object for one of the supported push notification serv
<code>delete_topic</code>	Deletes a topic and all its subscriptions
<code>get_endpoint_attributes</code>	Retrieves the endpoint attributes for a device on one of the supported push notificat
<code>get_platform_application_attributes</code>	Retrieves the attributes of the platform application object for the supported push no
<code>get_sms_attributes</code>	Returns the settings for sending SMS messages from your account
<code>get_subscription_attributes</code>	Returns all of the properties of a subscription
<code>get_topic_attributes</code>	Returns all of the properties of a topic
<code>list_endpoints_by_platform_application</code>	Lists the endpoints and endpoint attributes for devices in a supported push notificati
<code>list_phone_numbers_opted_out</code>	Returns a list of phone numbers that are opted out, meaning you cannot send SMS m
<code>list_platform_applications</code>	Lists the platform application objects for the supported push notification services, s
<code>list_subscriptions</code>	Returns a list of the requester's subscriptions
<code>list_subscriptions_by_topic</code>	Returns a list of the subscriptions to a specific topic
<code>list_tags_for_resource</code>	List all tags added to the specified Amazon SNS topic
<code>list_topics</code>	Returns a list of the requester's topics
<code>opt_in_phone_number</code>	Use this request to opt in a phone number that is opted out, which enables you to re
<code>publish</code>	Sends a message to an Amazon SNS topic, a text message (SMS message) directly
<code>remove_permission</code>	Removes a statement from a topic's access control policy
<code>set_endpoint_attributes</code>	Sets the attributes for an endpoint for a device on one of the supported push notifica
<code>set_platform_application_attributes</code>	Sets the attributes of the platform application object for the supported push notifica
<code>set_sms_attributes</code>	Use this request to set the default settings for sending SMS messages and receiving
<code>set_subscription_attributes</code>	Allows a subscription owner to set an attribute of the subscription to a new value
<code>set_topic_attributes</code>	Allows a topic owner to set an attribute of the topic to a new value
<code>subscribe</code>	Subscribes an endpoint to an Amazon SNS topic
<code>tag_resource</code>	Add tags to the specified Amazon SNS topic
<code>unsubscribe</code>	Deletes a subscription
<code>untag_resource</code>	Remove tags from the specified Amazon SNS topic

Examples

```
## Not run:
svc <- sns()
svc$add_permission(
  Foo = 123
)

## End(Not run)
```

`sqs`*Amazon Simple Queue Service*

Description

Welcome to the *Amazon Simple Queue Service API Reference*.

Amazon Simple Queue Service (Amazon SQS) is a reliable, highly-scalable hosted queue for storing messages as they travel between applications or microservices. Amazon SQS moves data between distributed application components and helps you decouple these components.

For information on the permissions you need to use this API, see [Identity and access management](#) in the *Amazon Simple Queue Service Developer Guide*.

You can use [AWS SDKs](#) to access Amazon SQS using your favorite programming language. The SDKs perform tasks such as the following automatically:

- Cryptographically sign your service requests
- Retry requests
- Handle error responses

Additional Information

- [Amazon SQS Product Page](#)
- *Amazon Simple Queue Service Developer Guide*
 - [Making API Requests](#)
 - [Amazon SQS Message Attributes](#)
 - [Amazon SQS Dead-Letter Queues](#)
- [Amazon SQS in the AWS CLI Command Reference](#)
- *Amazon Web Services General Reference*
 - [Regions and Endpoints](#)

Usage

```
sqs(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 <- sqs(
  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_permission	Adds a permission to a queue for a specific principal
change_message_visibility	Changes the visibility timeout of a specified message in a queue to a new value
change_message_visibility_batch	Changes the visibility timeout of multiple messages
create_queue	Creates a new standard or FIFO queue
delete_message	Deletes the specified message from the specified queue
delete_message_batch	Deletes up to ten messages from the specified queue
delete_queue	Deletes the queue specified by the QueueUrl, regardless of the queue's contents
get_queue_attributes	Gets attributes for the specified queue
get_queue_url	Returns the URL of an existing Amazon SQS queue
list_dead_letter_source_queues	Returns a list of your queues that have the RedrivePolicy queue attribute configured with a
list_queues	Returns a list of your queues in the current region
list_queue_tags	List all cost allocation tags added to the specified Amazon SQS queue
purge_queue	Deletes the messages in a queue specified by the QueueURL parameter
receive_message	Retrieves one or more messages (up to 10), from the specified queue
remove_permission	Revokes any permissions in the queue policy that matches the specified Label parameter
send_message	Delivers a message to the specified queue
send_message_batch	Delivers up to ten messages to the specified queue
set_queue_attributes	Sets the value of one or more queue attributes
tag_queue	Add cost allocation tags to the specified Amazon SQS queue
untag_queue	Remove cost allocation tags from the specified Amazon SQS queue

Examples

```

## Not run:
svc <- sqs()
svc$add_permission(
  Foo = 123
)

```

```
## End(Not run)
```

```
swf
```

Amazon Simple Workflow Service

Description

The Amazon Simple Workflow Service (Amazon SWF) makes it easy to build applications that use Amazon's cloud to coordinate work across distributed components. In Amazon SWF, a *task* represents a logical unit of work that is performed by a component of your workflow. Coordinating tasks in a workflow involves managing intertask dependencies, scheduling, and concurrency in accordance with the logical flow of the application.

Amazon SWF gives you full control over implementing tasks and coordinating them without worrying about underlying complexities such as tracking their progress and maintaining their state.

This documentation serves as reference only. For a broader overview of the Amazon SWF programming model, see the *Amazon SWF Developer Guide*.

Usage

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

Operations

<code>count_closed_workflow_executions</code>	Returns the number of closed workflow executions within the given domain that meet the filter
<code>count_open_workflow_executions</code>	Returns the number of open workflow executions within the given domain that meet the filter
<code>count_pending_activity_tasks</code>	Returns the estimated number of activity tasks in the specified task list
<code>count_pending_decision_tasks</code>	Returns the estimated number of decision tasks in the specified task list
<code>deprecate_activity_type</code>	Deprecates the specified activity type
<code>deprecate_domain</code>	Deprecates the specified domain
<code>deprecate_workflow_type</code>	Deprecates the specified workflow type
<code>describe_activity_type</code>	Returns information about the specified activity type
<code>describe_domain</code>	Returns information about the specified domain, including description and status
<code>describe_workflow_execution</code>	Returns information about the specified workflow execution including its type and some configuration settings
<code>describe_workflow_type</code>	Returns information about the specified workflow type
<code>get_workflow_execution_history</code>	Returns the history of the specified workflow execution
<code>list_activity_types</code>	Returns information about all activities registered in the specified domain that match the filter
<code>list_closed_workflow_executions</code>	Returns a list of closed workflow executions in the specified domain that meet the filter
<code>list_domains</code>	Returns the list of domains registered in the account
<code>list_open_workflow_executions</code>	Returns a list of open workflow executions in the specified domain that meet the filter
<code>list_tags_for_resource</code>	List tags for a given domain
<code>list_workflow_types</code>	Returns information about workflow types in the specified domain
<code>poll_for_activity_task</code>	Used by workers to get an ActivityTask from the specified activity taskList
<code>poll_for_decision_task</code>	Used by deciders to get a DecisionTask from the specified decision taskList
<code>record_activity_task_heartbeat</code>	Used by activity workers to report to the service that the ActivityTask represented by the taskToken is still running
<code>register_activity_type</code>	Registers a new activity type along with its configuration settings in the specified domain
<code>register_domain</code>	Registers a new domain
<code>register_workflow_type</code>	Registers a new workflow type and its configuration settings in the specified domain
<code>request_cancel_workflow_execution</code>	Records a WorkflowExecutionCancelRequested event in the currently running workflow execution history
<code>respond_activity_task_canceled</code>	Used by workers to tell the service that the ActivityTask identified by the taskToken was canceled
<code>respond_activity_task_completed</code>	Used by workers to tell the service that the ActivityTask identified by the taskToken completed
<code>respond_activity_task_failed</code>	Used by workers to tell the service that the ActivityTask identified by the taskToken has failed
<code>respond_decision_task_completed</code>	Used by deciders to tell the service that the DecisionTask identified by the taskToken has completed
<code>signal_workflow_execution</code>	Records a WorkflowExecutionSignaled event in the workflow execution history and creates a new taskList
<code>start_workflow_execution</code>	Starts an execution of the workflow type in the specified domain using the provided workflow type name
<code>tag_resource</code>	Add a tag to a Amazon SWF domain
<code>terminate_workflow_execution</code>	Records a WorkflowExecutionTerminated event and forces closure of the workflow execution
<code>undeprecate_activity_type</code>	Undeprecates a previously deprecated activity type
<code>undeprecate_domain</code>	Undeprecates a previously deprecated domain
<code>undeprecate_workflow_type</code>	Undeprecates a previously deprecated workflow type
<code>untag_resource</code>	Remove a tag from a Amazon SWF domain

Examples

```
## Not run:
svc <- swf()
svc$count_closed_workflow_executions(
  Foo = 123
)
```

End(Not run)

Index

activate_event_source, 3
add_permission, 9, 11

cancel_replay, 3
change_message_visibility, 11
change_message_visibility_batch, 11
check_if_phone_number_is_opted_out, 9
confirm_subscription, 9
count_closed_workflow_executions, 13
count_open_workflow_executions, 13
count_pending_activity_tasks, 13
count_pending_decision_tasks, 13
create_activity, 7
create_archive, 3
create_broker, 5
create_configuration, 5
create_event_bus, 3
create_partner_event_source, 3
create_platform_application, 9
create_platform_endpoint, 9
create_queue, 11
create_state_machine, 7
create_tags, 5
create_topic, 9
create_user, 5

deactivate_event_source, 3
delete_activity, 7
delete_archive, 3
delete_broker, 5
delete_endpoint, 9
delete_event_bus, 3
delete_message, 11
delete_message_batch, 11
delete_partner_event_source, 3
delete_platform_application, 9
delete_queue, 11
delete_rule, 3
delete_state_machine, 7
delete_tags, 5

delete_topic, 9
delete_user, 5
deprecate_activity_type, 13
deprecate_domain, 13
deprecate_workflow_type, 13
describe_activity, 7
describe_activity_type, 13
describe_archive, 3
describe_broker, 5
describe_broker_engine_types, 5
describe_broker_instance_options, 5
describe_configuration, 5
describe_configuration_revision, 5
describe_domain, 13
describe_event_bus, 3
describe_event_source, 3
describe_execution, 7
describe_partner_event_source, 3
describe_replay, 3
describe_rule, 3
describe_state_machine, 7
describe_state_machine_for_execution, 7
describe_user, 5
describe_workflow_execution, 13
describe_workflow_type, 13
disable_rule, 3

enable_rule, 3
eventbridge, 2

get_activity_task, 7
get_endpoint_attributes, 9
get_execution_history, 7
get_platform_application_attributes, 9
get_queue_attributes, 11
get_queue_url, 11
get_sms_attributes, 9
get_subscription_attributes, 9
get_topic_attributes, 9

get_workflow_execution_history, [13](#)
 list_activities, [7](#)
 list_activity_types, [13](#)
 list_archives, [3](#)
 list_brokers, [5](#)
 list_closed_workflow_executions, [13](#)
 list_configuration_revisions, [5](#)
 list_configurations, [5](#)
 list_dead_letter_source_queues, [11](#)
 list_domains, [13](#)
 list_endpoints_by_platform_application, [9](#)
 list_event_buses, [3](#)
 list_event_sources, [3](#)
 list_executions, [7](#)
 list_open_workflow_executions, [13](#)
 list_partner_event_source_accounts, [3](#)
 list_partner_event_sources, [3](#)
 list_phone_numbers_opted_out, [9](#)
 list_platform_applications, [9](#)
 list_queue_tags, [11](#)
 list_queues, [11](#)
 list_replays, [3](#)
 list_rule_names_by_target, [3](#)
 list_rules, [3](#)
 list_state_machines, [7](#)
 list_subscriptions, [9](#)
 list_subscriptions_by_topic, [9](#)
 list_tags, [5](#)
 list_tags_for_resource, [3, 7, 9, 13](#)
 list_targets_by_rule, [3](#)
 list_topics, [9](#)
 list_users, [5](#)
 list_workflow_types, [13](#)
 mq, [4](#)
 opt_in_phone_number, [9](#)
 poll_for_activity_task, [13](#)
 poll_for_decision_task, [13](#)
 publish, [9](#)
 purge_queue, [11](#)
 put_events, [3](#)
 put_partner_events, [3](#)
 put_permission, [4](#)
 put_rule, [4](#)
 put_targets, [4](#)
 reboot_broker, [5](#)
 receive_message, [11](#)
 record_activity_task_heartbeat, [13](#)
 register_activity_type, [13](#)
 register_domain, [13](#)
 register_workflow_type, [13](#)
 remove_permission, [4, 9, 11](#)
 remove_targets, [4](#)
 request_cancel_workflow_execution, [13](#)
 respond_activity_task_canceled, [13](#)
 respond_activity_task_completed, [13](#)
 respond_activity_task_failed, [13](#)
 respond_decision_task_completed, [13](#)
 send_message, [11](#)
 send_message_batch, [11](#)
 send_task_failure, [7](#)
 send_task_heartbeat, [7](#)
 send_task_success, [7](#)
 set_endpoint_attributes, [9](#)
 set_platform_application_attributes, [9](#)
 set_queue_attributes, [11](#)
 set_sms_attributes, [9](#)
 set_subscription_attributes, [9](#)
 set_topic_attributes, [9](#)
 sfn, [6](#)
 signal_workflow_execution, [13](#)
 sns, [8](#)
 sqs, [10](#)
 start_execution, [7](#)
 start_replay, [4](#)
 start_sync_execution, [7](#)
 start_workflow_execution, [13](#)
 stop_execution, [7](#)
 subscribe, [9](#)
 swf, [12](#)
 tag_queue, [11](#)
 tag_resource, [4, 7, 9, 13](#)
 terminate_workflow_execution, [13](#)
 test_event_pattern, [4](#)
 undeprecate_activity_type, [13](#)
 undeprecate_domain, [13](#)
 undeprecate_workflow_type, [13](#)
 unsubscribe, [9](#)
 untag_queue, [11](#)
 untag_resource, [4, 7, 9, 13](#)
 update_archive, [4](#)

update_broker, [5](#)
update_configuration, [5](#)
update_state_machine, [7](#)
update_user, [5](#)