Package ‘paws.application.integration’

January 12, 2020

Title Amazon Web Services Application Integration APIs

Version 0.1.6

Description Interface to Amazon Web Services application integration
   APIs, including 'Simple Queue Service' ('SQS') message queue, 'Simple
   Notification Service' ('SNS') publish/subscribe messaging, and more

License Apache License (>= 2.0)

Imports paws.common (>= 0.2.5)

Suggests testthat

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

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 2020-01-12 15:20:02 UTC

R topics documented:

   eventbridge ................................................................. 2
   mq ................................................................. 3
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.
```

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"
  )
)
```
mq

**Operations**

- activate_event_source: Activates a partner event source that has been deactivated
- 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: An AWS customer uses this operation to temporarily stop receiving events from the specified custom event bus or partner event bus
- 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_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 they have created
- describe_rule: Describes the specified rule
- disable_rule: Disables the specified rule
- enable_rule: Enables the specified rule
- list_event_buses: Lists all the event buses in your account, including the default event bus, custom event buses, and partner event buses
- 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 name is associated with
- list_partner_event_sources: An SaaS partner can use this operation to list all the partner event source names that they have created
- list_rule_names_by_target: Lists the rules for the specified target
- list_rules: Lists your 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 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 events to the specified event bus
- put_rule: Creates or updates the specified rule
- put_targets: Adds the specified targets to the specified rule, or updates the targets if they’re already associated
- remove_permission: Revokes the permission of another AWS account to be able to put events to the specified event bus
- remove_targets: Removes the specified targets from the specified rule
- 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

**Examples**

```r
svc <- eventbridge()
svc$activate_event_source(
  Foo = 123
)
```

**mq**

AmazonMQ
Description

Amazon MQ is a managed message broker service for Apache ActiveMQ 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.

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

create_broker Creates a broker
create_configuration Creates a new configuration for the specified configuration name
create_tags Add a tag to a resource
create_user Creates an ActiveMQ user
delete_broker Deletes a broker
delete_tags Removes a tag from a resource
delete_user Deletes an ActiveMQ user
describe_broker Returns information about the specified broker
describe_broker_engine_types Describe available engine types and versions
describe_broker_instance_options Describe available broker instance options
describe_configuration Returns information about the specified configuration
describe_configuration_revision Returns the specified configuration revision for the specified configuration
describe_user Returns information about an ActiveMQ user
list_brokers Returns a list of all brokers
list_configuration_revisions Returns a list of all revisions for the specified configuration
### 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](https://docs.aws.amazon.com/stepfunctions/latest/dg/).

### Usage

```r
sfn(config = list())
```

### Arguments

- **config**: Optional configuration of credentials, endpoint, and/or region.
Service syntax

```r
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 scheduled for execution by a running state machine
- `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 task identified by the taskToken failed
- `send_task_heartbeat` Used by activity workers and task states using the callback pattern to report to Step Functions that the task represented by the specified taskToken is still making progress
- `send_task_success` Used by activity workers and task states using the callback pattern to report that the task identified by the taskToken completed successfully
- `start_execution` Starts a 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 and/or roleArn

Examples

```r
svc <- sfn()
svc$create_activity()
```

```
Foo = 123
```
Amazon Simple Notification Service

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.

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.

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

add_permission
check_if_phone_number_is_opted_out
confirm_subscription
create_platform_application
create_platform_endpoint
create_topic
delete_endpoint
delete_platform_application
delete_topic
get_endpoint_attributes
get_platform_application_attributes
get_sms_attributes
get_subscription_attributes
get_topic_attributes
list_endpoints_by_platform_application
list_phone_numbers_opted_out
list_platform_application_attributes
list_platform_endpoints
list_platform_applications
list_subscriptions
list_subscriptions_by_topic
list_tags_for_resource
list_topics
opt_in_phone_number
publish
remove_permission
set_endpoint_attributes
set_platform_application_attributes
set_sms_attributes
set_subscription_attributes
set_topic_attributes
subscribe
tag_resource
unsubscribe
untag_resource

Adds a statement to a topic’s access control policy, granting access for the specified AWS accounts
Accepts a phone number and indicates whether the phone holder has opted out of receiving SMS messages
Verifies an endpoint owner’s intent to receive messages by validating the token sent to the endpoint
Creates a platform application object for one of the supported push notification services
Creates an endpoint for a device and mobile app on one of the supported push notification services
Creates a topic to which notifications can be published
Deletes the endpoint for a device and mobile app from Amazon SNS
Deletes a platform application object for one of the supported push notification services
Retrieves the endpoint attributes for a device on one of the supported push notification services
Retrieves the attributes of the platform application object for the supported push notification services
Returns the settings for sending SMS messages from your account
Returns all of the properties of a subscription
Returns all of the properties of a topic
Lists the endpoints and endpoint attributes for devices in a supported push notification service
Returns a list of phone numbers that are opted out, meaning you cannot send SMS messages to them
Lists the platform application objects for the supported push notification services, such as APNS and FCM
Returns a list of the requester’s subscriptions
Returns a list of the subscriptions to a specific topic
List all tags added to the specified Amazon SNS topic
Returns a list of the requester’s topics
Use this request to opt in a phone number that is opted out, which enables you to resume sending SMS messages
Sends a message to an Amazon SNS topic or sends a text message (SMS message)
Removes a statement from a topic’s access control policy
Sets the attributes for an endpoint for a device on one of the supported push notification services
Sets the attributes of the platform application object for the supported push notification services
Use this request to set the default settings for sending SMS messages and receiving daily SMS usage reports
Allows a subscription owner to set an attribute of the subscription to a new value
Allows a topic owner to set an attribute of the topic to a new value
Prepares to subscribe an endpoint by sending the endpoint a confirmation message
Add tags to the specified Amazon SNS topic
Deletes a subscription
Remove tags from the specified Amazon SNS topic

Examples

svc <- sns()
svc$add_permission(
  Foo = 123
)
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.

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

```python
sqs(config = list())
```

Arguments

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

Service syntax

```python
csvc <- 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
change_message_visibility
change_message_visibility_batch
create_queue
delete_message
delete_message_batch
delete_queue
get_queue_attributes
get_queue_url
list_dead_letter_source_queues
list_queues
list_queue_tags
purge_queue
receive_message
remove_permission
send_message
send_message_batch
set_queue_attributes
tag_queue
untag_queue

Adds a permission to a queue for a specific principal
Changes the visibility timeout of a specified message in a queue to a new value
Changes the visibility timeout of multiple messages
Creates a new standard or FIFO queue
Deletes the specified message from the specified queue
Deletes up to ten messages from the specified queue
Deletes the queue specified by the QueueUrl, regardless of the queue’s contents
Gets attributes for the specified queue
Returns the URL of an existing Amazon SQS queue
Returns a list of your queues that have the RedrivePolicy queue attribute configured with a
Returns a list of your queues
List all cost allocation tags added to the specified Amazon SQS queue
Deletes the messages in a queue specified by the QueueURL parameter
Retrieves one or more messages (up to 10), from the specified queue
Revokes any permissions in the queue policy that matches the specified Label parameter
Delivers a message to the specified queue
Delivers up to ten messages to the specified queue
Sets the value of one or more queue attributes
Add cost allocation tags to the specified Amazon SQS queue
Remove cost allocation tags from the specified Amazon SQS queue

Examples

svc <- sqs()
svc$add_permission(
  Foo = 123
)

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 wor-
rrying 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 pro-
gramming model, see the Amazon SWF Developer Guide.

Usage

```scala
swf(config = list())
```

Arguments

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

Service syntax

```scala
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

- `count_closed_workflow_executions` Returns the number of closed workflow executions within the given domain that meet the specified filter criteria.
- `count_open_workflow_executions` Returns the number of open workflow executions within the given domain that meet the specified filter criteria.
- `count_pending_activity_tasks` Returns the estimated number of activity tasks in the specified task list.
- `count_pending_decision_tasks` Returns the estimated number of decision tasks in the specified task list.
- `deprecate_activity_type` Deprecates the specified activity type.
- `deprecate_domain` Deprecates the specified domain.
- `deprecate_workflow_type` Deprecates the specified workflow type.
- `describe_activity_type` Returns information about the specified activity type.
- `describe_domain` Returns information about the specified domain, including description and status.
- `describe_workflow_execution` Returns information about the specified workflow execution including its type and some statistics.
- `describe_workflow_type` Returns information about the specified workflow type.
- `get_workflow_execution_history` Returns the history of the specified workflow execution.
- `list_activity_types` Returns information about all activities registered in the specified domain that match the specified filter criteria.
list_closed_workflow_executions
list_domains
list_open_workflow_executions
list_tags_for_resource
list_workflow_types
poll_for_activity_task
poll_for_decision_task
record_activity_task_heartbeat
register_activity_type
register_domain
register_workflow_type
request_cancel_workflow_execution
respond_activity_taskCanceled
respond_activity_task_completed
respond_activity_task_failed
respond_decision_task_completed
signal_workflow_execution
start_workflow_execution
tag_resource
terminate_workflow_execution
undeprecate_activity_type
undeprecate_domain
undeprecate_workflow_type
untag_resource

Returns a list of closed workflow executions in the specified domain that meet the filtering criteria.

Returns the list of domains registered in the account.

Returns a list of open workflow executions in the specified domain that meet the filtering criteria.

List tags for a given domain.

Returns information about workflow types in the specified domain.

Used by workers to get an ActivityTask from the specified activity taskList.

Used by activity workers to report to the service that the ActivityTask represented by the taskToken is still making progress.

Registers a new activity type along with its configuration settings in the specified domain.

Registers a new domain.

Registers a new workflow type and its configuration settings in the specified domain.

Records a WorkflowExecutionCanceled event in the currently running workflow execution.

Used by workers to tell the service that the ActivityTask identified by the taskToken was canceled.

Used by activity workers to report to the service that the ActivityTask represented by the taskToken completed.

Used by workers to tell the service that the ActivityTask identified by the taskToken completed.

Used by workers to tell the service that the ActivityTask identified by the taskToken failed.

Used by workers to report to the service that the ActivityTask represented by the taskToken failed.

Used by deciders to tell the service that the DecisionTask identified by the taskToken has completed.

Records a WorkflowExecutionSignaled event in the workflow execution history and creates a decision task.

Starts an execution of the workflow type in the specified domain using the provided workflowId and input data.

Add a tag to a Amazon SWF domain.

Records a WorkflowExecution Terminated event and forces closure of the workflow execution.

Undeprecates a previously deprecated activity type.

Undeprecates a previously deprecated domain.

Undeprecates a previously deprecated workflow type.

Remove a tag from a Amazon SWF domain.

Examples

```r
svc <- swf()
svc$count_closed_workflow_executions(
  Foo = 123
)
```
Index

activate_event_source, 3
add_permission, 8, 10
change_message_visibility, 10
count_closed_workflow_executions, 11
count_open_workflow_executions, 11
count_pending_activity_tasks, 11
count_pending_decision_tasks, 11
deactivate_event_source, 3
deactivate_activity_type, 11
deactivate_workflow_type, 11
dependent_domain, 11
dependent_workflow_type, 11
describe_activity, 6
describe_activity_type, 11
describe_broker, 4
describe_broker_engine_types, 4
describe_broker_instance_options, 4
describe_configuration, 4
describe_configuration_revision, 4
describe_configuration, 4
describe_configuration_revision, 4
describe_domain, 11
describe_event_bus, 3
describe_event_source, 3
describe_execution, 6
describe_partner_event_source, 3
describe_rule, 3
describe_state_machine, 6
describe_state_machine_for_execution, 6
describe_user, 4
describe_workflow_execution, 11
describe_workflow_type, 11
disable_rule, 3

delete_activity, 6
delete_broker, 4
delete_endpoint, 8
delete_event_bus, 3
delete_message, 10
delete_message_batch, 10
delete_partner_event_source, 3
delete_platform_application, 8
delete_queue, 10
delete_rule, 3
delete_state_machine, 6
delete_tags, 4
delete_topic, 8
delete_user, 4
deprecate_activity_type, 11

disable_activity, 6

describe_activity, 6
describe_task, 6

describe_activity, 6
describe_broker, 4
describe_broker_engine_types, 4
describe_broker_instance_options, 4
describe_configuration, 4
describe_configuration_revision, 4
describe_domain, 11
describe_event_bus, 3
describe_event_source, 3
describe_execution, 6
describe_partner_event_source, 3
describe_rule, 3
describe_state_machine, 6
describe_state_machine_for_execution, 6
describe_user, 4
describe_workflow_execution, 11
describe_workflow_type, 11
disable_rule, 3

delete_activity, 6
delete_broker, 4
delete_endpoint, 8
delete_event_bus, 3
delete_message, 10
delete_message_batch, 10
delete_partner_event_source, 3
delete_platform_application, 8
delete_queue, 10
delete_rule, 3
delete_state_machine, 6
delete_tags, 4
delete_topic, 8
delete_user, 4
deprecate_activity_type, 11

disable_activity, 6

describe_activity, 6
describe_task, 6

describe_activity, 6
describe_broker, 4
describe_broker_engine_types, 4
describe_broker_instance_options, 4
describe_configuration, 4
describe_configuration_revision, 4
describe_domain, 11
describe_event_bus, 3
describe_event_source, 3
describe_execution, 6
describe_partner_event_source, 3
describe_rule, 3
describe_state_machine, 6
describe_state_machine_for_execution, 6
describe_user, 4
describe_workflow_execution, 11
describe_workflow_type, 11
disable_rule, 3

delete_activity, 6
delete_broker, 4
delete_endpoint, 8
delete_event_bus, 3
delete_message, 10
delete_message_batch, 10
delete_partner_event_source, 3
delete_platform_application, 8
delete_queue, 10
delete_rule, 3
delete_state_machine, 6
delete_tags, 4
delete_topic, 8
delete_user, 4
deprecate_activity_type, 11

disable_activity, 6

describe_activity, 6
describe_task, 6

describe_activity, 6
describe_broker, 4
describe_broker_engine_types, 4
describe_broker_instance_options, 4
describe_configuration, 4
describe_configuration_revision, 4
describe_domain, 11
describe_event_bus, 3
describe_event_source, 3
describe_execution, 6
describe_partner_event_source, 3
describe_rule, 3
describe_state_machine, 6
describe_state_machine_for_execution, 6
describe_user, 4
describe_workflow_execution, 11
describe_workflow_type, 11
disable_rule, 3

delete_activity, 6
delete_broker, 4
delete_endpoint, 8
delete_event_bus, 3
delete_message, 10
delete_message_batch, 10
delete_partner_event_source, 3
delete_platform_application, 8
delete_queue, 10
delete_rule, 3
delete_state_machine, 6
delete_tags, 4
delete_topic, 8
delete_user, 4
deprecate_activity_type, 11

disable_activity, 6

describe_activity, 6
describe_task, 6

describe_activity, 6
describe_broker, 4
describe_broker_engine_types, 4
describe_broker_instance_options, 4
describe_configuration, 4
describe_configuration_revision, 4
describe_domain, 11
describe_event_bus, 3
describe_event_source, 3
describe_execution, 6
describe_partner_event_source, 3
describe_rule, 3
describe_state_machine, 6
describe_state_machine_for_execution, 6
describe_user, 4
describe_workflow_execution, 11
describe_workflow_type, 11
disable_rule, 3

delete_activity, 6
delete_broker, 4
delete_endpoint, 8
delete_event_bus, 3
delete_message, 10
delete_message_batch, 10
delete_partner_event_source, 3
delete_platform_application, 8
delete_queue, 10
delete_rule, 3
delete_state_machine, 6
delete_tags, 4
delete_topic, 8
delete_user, 4
deprecate_activity_type, 11

list_closed_workflow_executions, 12
list_configuration_revisions, 4
list_configurations, 5
list_dead_letter_source_queues, 10
list_domains, 12
list_endpoints_by_platform_application, 8
list_event_buses, 3
list_event_sources, 3
list_executions, 6
list_open_workflow_executions, 12
list_partner_event_source_accounts, 3
list_partner_event_sources, 3
list_phone_numbers_opted_out, 8
list_platform_applications, 8
list_queue_tags, 10
list_queues, 10
list_rule_names_by_target, 3
list_rules, 3
list_state_machines, 6
list_subscriptions, 8
list_subscriptions_by_topic, 8
list_tags, 5
list_tags_for_resource, 3, 6, 8, 12
list_targets_by_rule, 3
list_topics, 8
list_users, 5
list_workflow_types, 12
remove_permission, 3, 8, 10
remove_targets, 3
request_cancel_workflow_execution, 12
respond_activity_task_canceled, 12
respond_activity_task_completed, 12
respond_activity_task_failed, 12
respond_decision_task_completed, 12
send_message, 10
send_message_batch, 10
send_task_failure, 6
send_task_heartbeat, 6
send_task_success, 6
set_endpoint_attributes, 8
set_platform_application_attributes, 8
set_queue_attributes, 10
set_sms_attributes, 8
set_subscription_attributes, 8
set_topic_attributes, 8
sfn, 5
signal_workflow_execution, 12
sns, 7
sqs, 9
start_execution, 6
start_workflow_execution, 12
stop_execution, 6
subscribe, 8
swf, 10
tag_queue, 10
tag_resource, 3, 6, 8, 12
terminate_workflow_execution, 12
test_event_pattern, 3
undeprecate_activity_type, 12
undeprecate_domain, 12
undeprecate_workflow_type, 12
unsubscribe, 8
untag_queue, 10
untag_resource, 3, 6, 8, 12
update_broker, 5
documentation, 4
update_configuration, 5
update_state_machine, 6
update_user, 5

mq, 3
opt_in_phone_number, 8
poll_for_activity_task, 12
poll_for_decision_task, 12
publish, 8
purge_queue, 10
put_events, 3
put_partner_events, 3
put_permission, 3
put_rule, 3
put_targets, 3
reboot_broker, 5
receive_message, 10
record_activity_task_heartbeat, 12
register_activity_type, 12
register_domain, 12
register_workflow_type, 12