

# Package ‘paws.application.integration’

January 16, 2021

**Title** Amazon Web Services Application Integration Services

**Version** 0.1.10

**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

**LazyData** true

**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** 2021-01-16 21:20:02 UTC

## R topics documented:

eventbridge	2
mq	4
sfn	5
sns	7
sqs	9
swf	11

<b>Index</b>	<b>14</b>
--------------	-----------

---

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.

### 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

<a href="#">activate_event_source</a>	Activates a partner event source that has been deactivated
<a href="#">cancel_replay</a>	Cancels the specified replay
<a href="#">create_archive</a>	Creates an archive of events with the specified settings
<a href="#">create_event_bus</a>	Creates a new event bus within your account
<a href="#">create_partner_event_source</a>	Called by an SaaS partner to create a partner event source
<a href="#">deactivate_event_source</a>	You can use this operation to temporarily stop receiving events from the specified partner event source
<a href="#">delete_archive</a>	Deletes the specified archive
<a href="#">delete_event_bus</a>	Deletes the specified custom event bus or partner event bus
<a href="#">delete_partner_event_source</a>	This operation is used by SaaS partners to delete a partner event source
<a href="#">delete_rule</a>	Deletes the specified rule
<a href="#">describe_archive</a>	Retrieves details about an archive
<a href="#">describe_event_bus</a>	Displays details about an event bus in your account
<a href="#">describe_event_source</a>	This operation lists details about a partner event source that is shared with your account
<a href="#">describe_partner_event_source</a>	An SaaS partner can use this operation to list details about a partner event source that they have shared with your account
<a href="#">describe_replay</a>	Retrieves details about a replay
<a href="#">describe_rule</a>	Describes the specified rule
<a href="#">disable_rule</a>	Disables the specified rule
<a href="#">enable_rule</a>	Enables the specified rule
<a href="#">list_archives</a>	Lists your archives
<a href="#">list_event_buses</a>	Lists all the event buses in your account, including the default event bus, custom event bus, and partner event bus
<a href="#">list_event_sources</a>	You can use this to see all the partner event sources that have been shared with your AWS account
<a href="#">list_partner_event_source_accounts</a>	An SaaS partner can use this operation to display the AWS account ID that a particular partner event source is shared with
<a href="#">list_partner_event_sources</a>	An SaaS partner can use this operation to list all the partner event source names that they have shared with your account
<a href="#">list_replays</a>	Lists your replays
<a href="#">list_rule_names_by_target</a>	Lists the rules for the specified target
<a href="#">list_rules</a>	Lists your Amazon EventBridge rules
<a href="#">list_tags_for_resource</a>	Displays the tags associated with an EventBridge resource
<a href="#">list_targets_by_rule</a>	Lists the targets assigned to the specified rule
<a href="#">put_events</a>	Sends custom events to Amazon EventBridge so that they can be matched to rules
<a href="#">put_partner_events</a>	This is used by SaaS partners to write events to a customer's partner event bus
<a href="#">put_permission</a>	Running PutPermission permits the specified AWS account or AWS organization to put events to the specified partner event bus
<a href="#">put_rule</a>	Creates or updates the specified rule
<a href="#">put_targets</a>	Adds the specified targets to the specified rule, or updates the targets if they are already associated with the rule
<a href="#">remove_permission</a>	Revokes the permission of another AWS account to be able to put events to the specified partner event bus
<a href="#">remove_targets</a>	Removes the specified targets from the specified rule
<a href="#">start_replay</a>	Starts the specified replay
<a href="#">tag_resource</a>	Assigns one or more tags (key-value pairs) to the specified EventBridge resource
<a href="#">test_event_pattern</a>	Tests whether the specified event pattern matches the provided event
<a href="#">untag_resource</a>	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.

**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)
```

## 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.

## 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

<a href="#">create_activity</a>	Creates an activity
<a href="#">create_state_machine</a>	Creates a state machine
<a href="#">delete_activity</a>	Deletes an activity
<a href="#">delete_state_machine</a>	Deletes a state machine
<a href="#">describe_activity</a>	Describes an activity
<a href="#">describe_execution</a>	Describes an execution
<a href="#">describe_state_machine</a>	Describes a state machine
<a href="#">describe_state_machine_for_execution</a>	Describes the state machine associated with a specific execution
<a href="#">get_activity_task</a>	Used by workers to retrieve a task (with the specified activity ARN) which has been
<a href="#">get_execution_history</a>	Returns the history of the specified execution as a list of events
<a href="#">list_activities</a>	Lists the existing activities
<a href="#">list_executions</a>	Lists the executions of a state machine that meet the filtering criteria
<a href="#">list_state_machines</a>	Lists the existing state machines

<a href="#">list_tags_for_resource</a>	List tags for a given resource
<a href="#">send_task_failure</a>	Used by activity workers and task states using the callback pattern to report that the
<a href="#">send_task_heartbeat</a>	Used by activity workers and task states using the callback pattern to report to Step F
<a href="#">send_task_success</a>	Used by activity workers and task states using the callback pattern to report that the
<a href="#">start_execution</a>	Starts a state machine execution
<a href="#">start_sync_execution</a>	Starts a Synchronous Express state machine execution
<a href="#">stop_execution</a>	Stops an execution
<a href="#">tag_resource</a>	Add a tag to a Step Functions resource
<a href="#">untag_resource</a>	Remove a tag from a Step Functions resource
<a href="#">update_state_machine</a>	Updates an existing state machine by modifying its definition, roleArn, or loggingC

### Examples

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

## End(Not run)
```

---

 sns

*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](#).

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.

**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**

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



<a href="#">tag_resource</a>	Add tags to the specified Amazon SNS topic
<a href="#">unsubscribe</a>	Deletes a subscription
<a href="#">untag_resource</a>	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.

**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**

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

## Examples

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

## End(Not run)
```

## 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.

## 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 t
<code>count_open_workflow_executions</code>	Returns the number of open workflow executions within the given domain that meet th
<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 som
<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 th
<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 filterin
<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 th
<code>register_activity_type</code>	Registers a new activity type along with its configuration settings in the specified doma
<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
<code>respond_activity_task_canceled</code>	Used by workers to tell the service that the ActivityTask identified by the taskToken wa
<code>respond_activity_task_completed</code>	Used by workers to tell the service that the ActivityTask identified by the taskToken co
<code>respond_activity_task_failed</code>	Used by workers to tell the service that the ActivityTask identified by the taskToken ha
<code>respond_decision_task_completed</code>	Used by deciders to tell the service that the DecisionTask identified by the taskToken h
<code>signal_workflow_execution</code>	Records a WorkflowExecutionSignaled event in the workflow execution history and cre
<code>start_workflow_execution</code>	Starts an execution of the workflow type in the specified domain using the provided wo
<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 exe
<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, 8, 10  
  
cancel\_replay, 3  
change\_message\_visibility, 10  
change\_message\_visibility\_batch, 10  
check\_if\_phone\_number\_is\_opted\_out, 8  
confirm\_subscription, 8  
count\_closed\_workflow\_executions, 12  
count\_open\_workflow\_executions, 12  
count\_pending\_activity\_tasks, 12  
count\_pending\_decision\_tasks, 12  
create\_activity, 6  
create\_archive, 3  
create\_broker, 5  
create\_configuration, 5  
create\_event\_bus, 3  
create\_partner\_event\_source, 3  
create\_platform\_application, 8  
create\_platform\_endpoint, 8  
create\_queue, 10  
create\_state\_machine, 6  
create\_tags, 5  
create\_topic, 8  
create\_user, 5  
  
deactivate\_event\_source, 3  
delete\_activity, 6  
delete\_archive, 3  
delete\_broker, 5  
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, 5  
  
delete\_topic, 8  
delete\_user, 5  
deprecate\_activity\_type, 12  
deprecate\_domain, 12  
deprecate\_workflow\_type, 12  
describe\_activity, 6  
describe\_activity\_type, 12  
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, 12  
describe\_event\_bus, 3  
describe\_event\_source, 3  
describe\_execution, 6  
describe\_partner\_event\_source, 3  
describe\_replay, 3  
describe\_rule, 3  
describe\_state\_machine, 6  
describe\_state\_machine\_for\_execution, 6  
describe\_user, 5  
describe\_workflow\_execution, 12  
describe\_workflow\_type, 12  
disable\_rule, 3  
  
enable\_rule, 3  
eventbridge, 2  
  
get\_activity\_task, 6  
get\_endpoint\_attributes, 8  
get\_execution\_history, 6  
get\_platform\_application\_attributes, 8  
get\_queue\_attributes, 10  
get\_queue\_url, 10  
get\_sms\_attributes, 8  
get\_subscription\_attributes, 8  
get\_topic\_attributes, 8

get\_workflow\_execution\_history, [12](#)

list\_activities, [6](#)

list\_activity\_types, [12](#)

list\_archives, [3](#)

list\_brokers, [5](#)

list\_closed\_workflow\_executions, [12](#)

list\_configuration\_revisions, [5](#)

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\_replays, [3](#)

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](#), [7](#), [8](#), [12](#)

list\_targets\_by\_rule, [3](#)

list\_topics, [8](#)

list\_users, [5](#)

list\_workflow\_types, [12](#)

mq, [4](#)

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](#)

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, [7](#)

send\_task\_heartbeat, [7](#)

send\_task\_success, [7](#)

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, [7](#)

start\_replay, [3](#)

start\_sync\_execution, [7](#)

start\_workflow\_execution, [12](#)

stop\_execution, [7](#)

subscribe, [8](#)

swf, [11](#)

tag\_queue, [10](#)

tag\_resource, [3](#), [7](#), [9](#), [12](#)

terminate\_workflow\_execution, [12](#)

test\_event\_pattern, [3](#)

undeprecate\_activity\_type, [12](#)

undeprecate\_domain, [12](#)

undeprecate\_workflow\_type, [12](#)

unsubscribe, [9](#)

untag\_queue, [10](#)

untag\_resource, [3](#), [7](#), [9](#), [12](#)

update\_archive, [4](#)

update\_broker, [5](#)  
update\_configuration, [5](#)  
update\_state\_machine, [7](#)  
update\_user, [5](#)