Package ‘paws.application.integration’

March 9, 2021

Title Amazon Web Services Application Integration Services

Version 0.1.11

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 2021-03-09 14:20:02 UTC

R topics documented:

`eventbridge` *Amazon EventBridge*

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

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 partn
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 th
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 b
list_event_sources	You can use this to see all the partner event sources that have been shared with your AW
list_partner_event_source_accounts	An SaaS partner can use this operation to display the AWS account ID that a particular
list_partner_event_sources	An SaaS partner can use this operation to list all the partner event source names that the
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.

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
list_configurations	Returns a list of all configurations
list_tags	Lists tags for a resource
list_users	Returns a list of all ActiveMQ users
reboot_broker	Reboots a broker
update_broker	Adds a pending configuration change to a broker
update_configuration	Updates the specified configuration
update_user	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.

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 t
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 t
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)
```

---

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

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

tag_resource	Add tags to the specified Amazon SNS topic
unsubscribe	Deletes a subscription
untag_resource	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

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
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.

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

count_closed_workflow_executions	Returns the number of closed workflow executions within the given domain that meet t
count_open_workflow_executions	Returns the number of open workflow executions within the given domain that meet th
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 som
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 th
list_closed_workflow_executions	Returns a list of closed workflow executions in the specified domain that meet the filter
list_domains	Returns the list of domains registered in the account
list_open_workflow_executions	Returns a list of open workflow executions in the specified domain that meet the filterin
list_tags_for_resource	List tags for a given domain
list_workflow_types	Returns information about workflow types in the specified domain
poll_for_activity_task	Used by workers to get an ActivityTask from the specified activity taskList
poll_for_decision_task	Used by deciders to get a DecisionTask from the specified decision taskList
record_activity_task_heartbeat	Used by activity workers to report to the service that the ActivityTask represented by th
register_activity_type	Registers a new activity type along with its configuration settings in the specified doma
register_domain	Registers a new domain
register_workflow_type	Registers a new workflow type and its configuration settings in the specified domain
request_cancel_workflow_execution	Records a WorkflowExecutionCancelRequested event in the currently running workflo
respond_activity_task_canceled	Used by workers to tell the service that the ActivityTask identified by the taskToken wa
respond_activity_task_completed	Used by workers to tell the service that the ActivityTask identified by the taskToken co
respond_activity_task_failed	Used by workers to tell the service that the ActivityTask identified by the taskToken ha
respond_decision_task_completed	Used by deciders to tell the service that the DecisionTask identified by the taskToken h
signal_workflow_execution	Records a WorkflowExecutionSignaled event in the workflow execution history and cre
start_workflow_execution	Starts an execution of the workflow type in the specified domain using the provided wo
tag_resource	Add a tag to a Amazon SWF domain
terminate_workflow_execution	Records a WorkflowExecutionTerminated event and forces closure of the workflow exe
undeprecate_activity_type	Undeprecates a previously deprecated activity type
undeprecate_domain	Undeprecates a previously deprecated domain
undeprecate_workflow_type	Undeprecates a previously deprecated workflow type
untag_resource	Remove a tag from a Amazon SWF domain

Examples

```
## Not run:
```

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

## End(Not run)
```

Index