SGCWEBSOCKETS · TECHNICAL DOCUMENT

Amazon SQS Client

Amazon SQS HTTP client — send, receive, delete and manage SQS messages from Delphi with AWS Signature v4.

eSeGeCe

WWW.ESEGECE.COM

Overview

Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS eliminates the complexity and overhead associated with managing and operating message oriented middleware, and empowers developers to focus on differentiating work. Using SQS, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available.

At a glance

COMPONENT CLASS

`TsgcHTTPAWS_SQS_Client`

STANDARDS / SPEC

Amazon SQS — Developer Guide

TRANSPORTS

TCP, TLS

PLATFORMS

Windows, macOS, Linux, iOS, Android

FRAMEWORKS

VCL, FireMonkey, Lazarus / FPC

EDITION

Standard / Professional / Enterprise

Features

- Native Delphi implementation with full ANSI/Unicode support.

Technical specification

Standards & specs	Amazon SQS — Developer Guide · Amazon SQS — API reference
Component class	`TsgcHTTPAWS_SQS_Client` (unit `sgcHTTP_AWS_SQS_Client`)
Frameworks	VCL, FireMonkey, Lazarus / FPC
Platforms	Windows, macOS, Linux, iOS, Android

Main properties

The principal published / public properties used to configure and drive the component. Consult the online help for the full list.

`EndPoint`	Published or public property used to configure or query the component.
`AWSOptions`	Published or public property used to configure or query the component.
`LogFile`	Published or public property used to configure or query the component.
`OnSQSBeforeRequest`	Published or public property used to configure or query the component.
`OnSQSResponse`	Published or public property used to configure or query the component.
`OnSQSError`	Published or public property used to configure or query the component.
`Version`	Published or public property used to configure or query the component.

Main methods

The principal public methods exposed by the component.

`SendMessageBatch()`	Public procedure exposed by the component.
`DeleteMessageBatch()`	Public procedure exposed by the component.
`ReceiveMessage()`	Public function exposed by the component.
`DeleteMessage()`	Public function exposed by the component.
`DeleteQueue()`	Public function exposed by the component.

`PostURL()`	Public function exposed by the component.
`ChangeMessageVisibilityBatch()`	Public function exposed by the component.
`ListQueues()`	Public function exposed by the component.
`ListQueueTags()`	Public function exposed by the component.
`ListDeadLetterSourceQueues()`	Public function exposed by the component.

Quick Start

Drop the component on a form, configure the properties below and activate it. The snippet that follows shows the typical **SQS Client** configuration sourced from the online help.

> **About this scenario.** Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS eliminates the complexity and overhead associated with managing and operating message oriented middleware, and empowers developers to focus on differentiating work. Using SQS, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available.

Delphi (VCL / FireMonkey)

```
// <b>TsgcHTTPAWS_SQS_Client</b> is the component used for connect to Amazon SQS.
// Client connects using HTTPs protocol and authenticates using Access Key provided by Amazon.

// Before you attempt to connect to SQS service, you must set some data in <b>AWSOptions</b> pro

// <b>Region:</b> your endpoint region, example: us-east-1.
// <b>AccessKey:</b> access key provided by Amazon.
// <b>SecretKey:</b> secret key provided by Amazon.

// The following methods are supported by SQS client:

// <b>AddPermission</b>
// Adds a permission to a queue for a specific principal. This allows sharing access to the queu

// <b>ChangeMessageVisibility</b>
// Changes the visibility timeout of a specified message in a queue to a new value. The default
// The minimum is 0 seconds. The maximum is 12 hours.

// <b>ChangeMessageVisibilityBatch</b>

// Changes the visibility timeout of multiple messages. This is a batch version of ChangeMessage
// The result of the action on each message is reported individually in the response. You can se

// <b>CreateQueue</b>
// Creates a new standard or FIFO queue. You can pass one or more attributes in the request.

  vURL := SQS.CreateQueue('sqs_queue');
  if vURL  '' then
    DoLog('#CreateQueue: ' + vURL);

// <b>DeleteMessage</b>
// Deletes the specified message from the specified queue. To select the message to delete, use

  if SQS.DeleteMessage('sqs_queue', '...receipt handle goes here...') then
    DoLog('#DeleteMessage: ok');
  else
    DoLog('#DeleteMessage: error');

// <b>DeleteMessageBatch</b>
// Deletes up to ten messages from the specified queue. This is a batch version of DeleteMessage

// <b>DeleteQueue</b>
// Deletes the queue specified by the queue name, regardless of the queue's contents.

  if SQS.DeleteQueue(txtQueueName.Text) then
    DoLog('#Delete Queue: ok')
  else
    DoLog('#Delete Queue: error');

// <b>GetQueueAttributes</b>
// Gets attributes for the specified queue.
```

```
  oAttributes := TsgcSQSAttributes.Create;
  Try
    if SQS.GetQueueAttributes('sqs_queue', oAttributes) then
    begin
      for i := 0 to oAttributes.Count - 1 do
        DoLog('#Attribute: ' + TsgcSQSAttribute(oAttributes→Item[i])
          .AttributeName + ' ' + TsgcSQSAttribute(oAttributes→Item[i])
          .AttributeValue);
    end
    else
      DoLog('#GetQueueAttributes: error');
  Finally
    FreeAndNil(oAttributes);
  End;

// <b>GetQueueUrl</b>
// Returns the URL of an existing Amazon SQS queue.

// <b>ListDeadLetterSourceQueues</b>
// Returns a list of your queues that have the RedrivePolicy queue attribute configured with a d

// <b>ListQueueTags</b>
// List all cost allocation tags added to the specified Amazon SQS queue.

// <b>PurgeQueue</b>
// Deletes the messages in a queue specified by the QueueName parameter.

  if SQS.PurgueQueue('sqs_queue') then
    DoLog('#PurgueQueue: ok')
  else
    DoLog('#PurgueQueue: error');

// <b>ReceiveMessage</b>
// Retrieves one or more messages (up to 10), from the specified queue.

  oResponses := TsgcSQSReceiveMessageResponses.Create;
  Try
    if SQS.ReceiveMessage('sqs_test', oResponses) then
    begin
      for i := 0 to oResponses.Count - 1 do
      begin
        DoLog('#ReceiveMessage: ' + TsgcSQSReceiveMessageResponse(oResponses.Item[i]).Body);
        FReceiptHandle := TsgcSQSReceiveMessageResponse(oResponses.Item[i]).ReceiptHandle;
      end;
    end;
  Finally
    FreeAndNil(oResponses);
  End;

// <b>RemovePermission</b>
// Revokes any permissions in the queue policy that matches the specified Label parameter.

// <b>SendMessage</b>
// Delivers a message to the specified queue.
```

```
  if SQS.SendMessage('sqs_queue', 'My First Message') then
    DoLog('#SendMessage: ok')
  else
    DoLog('#SendMessage: error');

// <b>SendMessageBatch</b>
// Delivers up to ten messages to the specified queue. This is a batch version of SendMessage.

// <b>SetQueueAttributes</b>
// Sets the value of one or more queue attributes. When you change a queue's attributes, the cha
// for most of the attributes to propagate throughout the Amazon SQS system.

  oAttributes := TsgcSQSAttributes.Create;
  Try
    oAttributes.AddSQSAttribute(sqsatVisibilityTimeout, '45');
    if SQS.SetQueueAttributes('sqs_queue', oAttributes) then
      DoLog('#SetQueueAttributes: ok')
    else
      DoLog('#SetQueueAttributes: error');
  Finally
    FreeAndNil(oAttributes);
  End;

// <b>TagQueue</b>
// Add cost allocation tags to the specified Amazon SQS queue.

// <b>UntagQueue</b>
// Remove cost allocation tags from the specified Amazon SQS queue.
```

C++ Builder

```
// <b>TsgcHTTPAWS_SQS_Client</b> is the component used for connect to Amazon SQS.
// Client connects using HTTPs protocol and authenticates using Access Key provided by Amazon.

// Before you attempt to connect to SQS service, you must set some data in <b>AWSOptions</b> pro

// <b>Region:</b> your endpoint region, example: us-east-1.
// <b>AccessKey:</b> access key provided by Amazon.
// <b>SecretKey:</b> secret key provided by Amazon.

// The following methods are supported by SQS client:

// <b>AddPermission</b>
// Adds a permission to a queue for a specific principal. This allows sharing access to the queu

// <b>ChangeMessageVisibility</b>
// Changes the visibility timeout of a specified message in a queue to a new value. The default
// The minimum is 0 seconds. The maximum is 12 hours.

// <b>ChangeMessageVisibilityBatch</b>

// Changes the visibility timeout of multiple messages. This is a batch version of ChangeMessage
// The result of the action on each message is reported individually in the response. You can se

// <b>CreateQueue</b>
// Creates a new standard or FIFO queue. You can pass one or more attributes in the request.

  vURL = SQS->CreateQueue("sqs_queue");
  if (vURL != "")
  {
    DoLog("#CreateQueue: " + vURL);
  }

// <b>DeleteMessage</b>
// Deletes the specified message from the specified queue. To select the message to delete, use

  if (SQS->DeleteMessage("sqs_queue", "...receipt handle goes here...") == true)
  {
    DoLog("#DeleteMessage: ok");
  }
  else
  {
    DoLog("#DeleteMessage: error");
  }

// <b>DeleteMessageBatch</b>
// Deletes up to ten messages from the specified queue. This is a batch version of DeleteMessage

// <b>DeleteQueue</b>
// Deletes the queue specified by the queue name, regardless of the queue's contents.

  if (SQS->DeleteQueue(txtQueueName->Text) == true)
  {
```

```
    DoLog("#Delete Queue: ok");
  }
  else
  {
    DoLog("#Delete Queue: error");
  }

// <b>GetQueueAttributes</b>
// Gets attributes for the specified queue.

  TsgcSQSAttributes *oAttributes = new TsgcSQSAttributes();
  try
  {
    if (SQS->GetQueueAttributes("sqs_queue", oAttributes) == true)
    {
      for (int i = 0; i < oAttributes->Count; i++)
      {
        DoLog("#Attribute: " + static_cast<TsgcSQSAttribute*>(oAttributes->Item[i])
          ->AttributeName + " " + static_cast<TsgcSQSAttribute*>(oAttributes->Item[i])
          ->AttributeValue);
      }
    }
    else
    {
      DoLog("#GetQueueAttributes: error");
    }
  }
  __finally
  {
      oAttributes->Free();
  }

// <b>GetQueueUrl</b>
// Returns the URL of an existing Amazon SQS queue.

// <b>ListDeadLetterSourceQueues</b>
// Returns a list of your queues that have the RedrivePolicy queue attribute configured with a d

// <b>ListQueueTags</b>
// List all cost allocation tags added to the specified Amazon SQS queue.

// <b>PurgeQueue</b>
// Deletes the messages in a queue specified by the QueueName parameter.

  if (SQS->PurgueQueue("sqs_queue") == true)
  {
    DoLog("#PurgueQueue: ok");
  }
  else
  {
    DoLog("#PurgueQueue: error");
  }

// <b>ReceiveMessage</b>
// Retrieves one or more messages (up to 10), from the specified queue.
```

```
  TsgcSQSReceiveMessageResponses *oResponses = new TsgcSQSReceiveMessageResponses();
  try
  {
    if (SQS->ReceiveMessage("sqs_test", oResponses) == true)
    {
      for (int i = 0; i < oResponses->Count; i++)
      {
        DoLog("#ReceiveMessage: " + static_cast<TsgcSQSReceiveMessageResponse*>(oResponses->Item
        FReceiptHandle = static_cast<TsgcSQSReceiveMessageResponse*>(oResponses->Item[i])->Recei
      }
    }
  }
  __finally
  {
    oResponses->Free();
  }

// <b>RemovePermission</b>
// Revokes any permissions in the queue policy that matches the specified Label parameter.

// <b>SendMessage</b>
// Delivers a message to the specified queue.

  if (SQS->SendMessage("sqs_queue", "My First Message") == true)
  {
    DoLog("#SendMessage: ok");
  }
  else
  {
    DoLog("#SendMessage: error");
  }

// <b>SendMessageBatch</b>
// Delivers up to ten messages to the specified queue. This is a batch version of SendMessage.

// <b>SetQueueAttributes</b>
// Sets the value of one or more queue attributes. When you change a queue's attributes, the cha
// for most of the attributes to propagate throughout the Amazon SQS system.

  TsgcSQSAttributes *oAttributes = new TsgcSQSAttributes();
  try
  {
    oAttributes->AddSQSAttribute(sqsatVisibilityTimeout, "45");
    if (SQS->SetQueueAttributes("sqs_queue", oAttributes) == true)
    {
      DoLog("#SetQueueAttributes: ok");
    }
    else
    {
      DoLog("#SetQueueAttributes: error");
    }
  }
  __finally
  {
```

```
    oAttributes->Free();
  }

// <b>TagQueue</b>
// Add cost allocation tags to the specified Amazon SQS queue.

// <b>UntagQueue</b>
// Remove cost allocation tags from the specified Amazon SQS queue.
```

.NET (C#)

```
```

Sources used to build this document

Every external claim links back to a primary source. The online-help references decode the canonical deep-link the company maintains for this component.

Primary standard / spec — Amazon SQS — Developer Guide	docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html
Primary standard / spec — Amazon SQS — API reference	docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/Welcome.html
Online help — component page	www.esegece.com/help/sgcWebSockets/Components/HTTP/Amazon/Amazon_SQS.htm
Delphi demo project (in the sgcWebSockets package)	`Demos\20.HTTP_Protocol\04.AWS\01.Amazon_SQS`
Component page	www.esegece.com/products/websockets/http/amazon-sqs/
Product page	www.esegece.com/products/websockets/

> **Document scope.** This document covers the publicly-documented surface of the Amazon SQS Client component shipped with sgcWebSockets. For full property, method and event reference consult the online help linked above.