AMAZON SQS

A FULLY-MANAGED MESSAGE QUEUING SERVICE IN THE CLOUD

INTRODUCTION 👋

Amazon **S**imple **Q**ueue **S**ervice (**SQS**) is a fully-managed message queuing service that drastically helps to decouple systems, increase reliability, scale microservices, and serverless applications.

Believe it or not: SQS was the **first** publicly launched service by AWS.

Quoting **Jeff Bar**:
"We launched the **Simple Queue Service** in **late 2004**, **Amazon S3** in early 2006, and **Amazon EC2** later that summer."

One more detail to get you excited about **SQS's capabilities**: At Amazon's Prime Days in 2021, a new traffic record was set with SQS processing **47.7 million** ✉ **messages per second** at its peak. 🔥

COMMON RECOMMENDATIONS 📚

A few points to focus on when starting with SQS:

- **Proper configuration of the visibility timeout** - make sure visibility timeouts are not too high (re-processing of messages can take a long time) or too low (high risk for duplicate message processing)
- **Capturing unprocessable messages** - redrive messages to a DLQ to decrease chances of being exposed to **poison pill messages** (received but unprocessable messages that are kept) & the ability to easily investigate issues.
- **Proper Retention for DLQs** - the retenion period of the DLQ includes the time the message spends in the original queue.

- **Setting up long polling** - helps to reduce costs due to a reduced number of empty responses.
- **Handling Request Errors** - make proper use of retry and backoff logics. The AWS SDK offers this out of the box.

EVENT-DRIVEN ARCHITECTURES 📩

Event-driven architectures are a very common way of realizing modern microservice applications. They are built around events which are exchanged between components and either carry a state or used as identifiers.

A message will be produced by one component (**Producer**) and consumed by another (**Consumer**). This enables easier error-handling (e.g. reprocessing of sub-routines) which will result in a less coupled and error-prone system.

Even more: it's not fatal if...

- the producer is working faster than the consumer or
- the producer or consumer has only intermittent network connection

... as the message queue **can** shrink or increase at different speeds without causing harm or side effects.

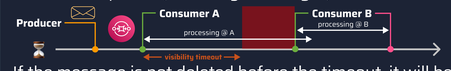
QUEUE TYPES 🌟

There are several types of queues for varying requirements:

- **Standard** - the default, with nearly unlimited API calls per second. Messages may be delivered out of order.
- **FIFO** - **F**irst **I**n, **F**irst **O**ut; messages are delivered in strict order (relative to their message group, if there are any).
- **Delay** - postpone the delivery of new messages for consumers for a number of seconds.
- **Temporary** - create a queue on demand & clean it up when it's not of use anymore.

VISIBILITY TIMEOUTS 🔴

After receiving a message, it's hidden for other consumers for a certain period (= message is **inflight**).



If the message is not deleted before the timeout, it will be available again which could result in messages being delivered & processed **more than once**.

💡 Default timeout is **30 seconds** & max timeout is **12 hours**.

QUEUE PARAMETERS 📝

Besides **Visibility Timeou**t & **Retention Period**, there are additional configurations for queues you can take:

- **Delivery Delay** - how long should SQS wait until delivering a message that has been added to the queue.
- **Receive Message wait time** - timeout until SQS sends an emty response for a long poll if there are no queued messages available.
- **Content-based Deduplication** - enable if you want SQS to create unique deduplication IDs based on the message content.
- **Redrive Allow Policy** - configure which source queues are allowed to use this queue as a DLQ.

ACCESS POLICIES 🔑

As with all other services, you can configure policies that define the accounts, users, and roles that can access your queues and which actions are **allowed** or **denied**.

RETENTION PERIODS 💾

Define how long SQS stores each individual message while waiting for it to be processed and deleted. It can range from **1 minute** up to **14 days**.

SQS will send messages to the configured DLQ if there is any

If the retention period is over, the message will be automatically **discarded** from the queue.

LAMBDA TRIGGERS λ

You can process SQS messages via a Lambda function. Lambda will poll the queue & invoke your function synchronously with an event that contains the queue message.

💡 Messages for Lambda triggers can be aggregated together into **batches.** By that, a single function invocation processes **several messages** at a time. This saves compute time, as there are fewer cold starts on average.

ENCRYPTION 🔐

You can configure **S**erver-**S**ide **E**ncryption (**SSE**) either via SQS-owned encryption keys (**SSE-SQS**) or via keys that are managed within AWS Key Management Service (**SSE-KMS**).

Both encryption modes protected messages at rest using 256-bit AES encryption. The messages will be **encrypted as soon as SQS recieves** them & **only decrypted when sending them to authorized consumers**.

💡 If using SSE-KMS, make sure all of your queue's principals have sufficient permissions for the encryption key.

LIMITATIONS 🔶

As with other services, everything has its limitations:

- **120k messages can be inflight** at a time - for FIFO Queues it's only 20k.
- **256 KB message size** limit (there are library extensions for SQS to store messages in S3 & only send references).
- FIFO "only" allows **300 operations per second**.

The last limitation can be tackled by using **message batching** to receive up to 10 messages at once, which will therefore result in being able to process up to 3,000 messages per second per queue.

MESSAGE POLLING ⌛

SQS provides two different types of message polling:

- **Short Polling** - active by default; a **ReceiveMessage** request is only sent to a subset of servers to find available messages. Even if there's no available message, the response is sent **immediately**.
- **Long Polling** - the ReceiveMessage request will query **all servers** for available messages. SQS will wait until at least one message is available or the polling wait time expires before sending a response.

DEAD-LETTER QUEUES ☠

You need to define how many times a message can be retrieved from a consumer until it is considered unprocessable and will be discarded from the queue.

Instead of discarding the message: define another queue as a **D**ead-**L**etter **Q**ueue (**DLQ**) to forward messages to after the limit is exceeded.

Dead Letter Queues help you to unblock your messaging systems, **without** losing messages. If a message is unprocessable, you can either define an automated exception handling process or just step in for taking manual diagnostics to solve the underlying issue.

MESSAGE METADATA ⌗

Each SQS message can contain up to 10 custom **metadata attributes**. These attributes are delivered together with the message body.

Consumers can use message attributes to handle messages in a particular way **without having to processes the entire message body first**.

MONITORING 🌱

By default, SQS will frequently send metrics to CloudWatch like the **visible messages count**, the **number of empty receives**, **messages that had been sent to the queue**, or the **number of deleted messages**.

One-click integration tools like **Dashbird.io** help to visualize all those metrics with a **all-in-one dashboard** so you always know what's up. Additionally, you get **well-architected hints** for best practices.

aws