Azure Queue Service

What is Azure Queue Service?

Azure Queue Service is part of **Azure Storage**. It provides cloud-based messaging that allows communication between application components through **asynchronous messages** stored in queues.

It is designed for **simple queuing scenarios**, where producers send messages and consumers retrieve and process them independently.

Technical Specifications

Feature	Details
Message Size	Up to 64 KB (UTF-8 text or Base64-encoded binary)
Maximum Queue Size	Up to 500 TB (based on storage account limits)
Message TTL (Time-to-Live)	7 days (configurable per message)
Delivery Guarantee	At-least-once
Access Protocols	HTTP/HTTPS (REST API)
Authentication	Shared access signature (SAS) tokens or Azure AD
Latency	Low latency (~milliseconds)
Pricing	Based on storage and number of operations

Architecture & Workflow

1. **Producer** writes a message to the queue.
2. **Queue Storage** stores the message durably.
3. **Consumer (worker role, function app, or WebJob)** reads and processes the message.
4. Once processed, the message is **explicitly deleted** from the queue.

   If not deleted within the visibility timeout, the message becomes visible again for reprocessing (potential duplicate).

Ideal Use Cases

- Background job processing (e.g., image resizing)
- Decoupling web frontends from backend processors
- Throttling workload (buffer burst traffic)
- Simplified batch processing

Example Use Case

Scenario: An e-commerce website uploads customer images to a blob storage. It sends a message to the queue with image metadata. A background worker polls the queue, processes the image (e.g., resize or format conversion), and updates the database once complete.

Azure Service Bus

What is Azure Service Bus?

Azure Service Bus is a **fully managed enterprise messaging platform** capable of handling complex messaging scenarios including **message queuing**, **publish-subscribe patterns**, **message ordering**, **transactional processing**, and **dead-lettering**.

It is suitable for **enterprise-grade applications** requiring reliability, state management, and advanced messaging patterns.

Technical Specifications

Feature	Details
Message Size	Up to 256 KB (Standard tier), 1 MB (Premium tier)
Entities	Queues, Topics, Subscriptions
Protocols	AMQP, HTTP/HTTPS
Features	Dead-lettering, Duplicate detection, Message sessions, Auto-forwarding, Scheduled delivery, Transactions
Delivery Guarantees	At-least-once or Exactly-once (via sessions and transactions)
Authentication	Shared Access Policies or Azure AD
Geo-redundancy	Supported (Premium tier)
Pricing	Based on messaging operations and tier (Standard or Premium)

Architecture & Messaging Patterns

- **Point-to-point (Queue):** One sender, one receiver.
- **Publish-subscribe (Topic + Subscriptions):** One sender, multiple subscribers.
- **Sessions:** Maintain stateful message processing across a session ID.
- **Dead-letter Queues (DLQs):** Handle messages that cannot be delivered or processed.
- **Auto-forwarding:** Automatically transfer messages between entities.

Ideal Use Cases

- Event-driven architectures
- Microservices communication
- Business process orchestration (e.g., order pipelines)
- Financial systems requiring transactional messaging
- Scheduled message processing (e.g., send reminders 24 hours later)

Example Use Case

Scenario: A banking application uses Service Bus to process loan applications. Each application triggers a message sent to a Topic. Subscriptions include:

- Fraud Check Service
- Credit Score Evaluator
- Document Verifier

Each service processes the message independently, and the result is aggregated for approval. Failed or unprocessed messages are sent to a dead-letter queue for manual review.

When to Use What

Use Azure Queue Service if:

- Your needs are **simple and cost-sensitive**.
- You're building a **lightweight workload** that doesn't require ordering or pub-sub.
- Your application can tolerate **duplicate or out-of-order messages**.
- You want to integrate with **Azure Functions** easily.

Real-Life Example:

> A travel website processes uploaded booking PDFs. Each upload triggers a message in Azure Queue Service. A worker reads the message, parses the PDF, and stores structured data in a database.

Use Azure Service Bus if:

- You require **advanced messaging features**.
- Your application is **mission-critical** and cannot lose or misorder messages.
- You need **message routing**, **multiple subscribers**, or **transactional workflows**.
- You're building a **microservices architecture** or enterprise system.

Real-Life Example:

> A healthcare platform routes patient intake forms to multiple processing systems — insurance validation, appointment scheduling, and patient profile creation — using Service Bus Topics and Subscriptions. Dead-lettering ensures failed messages are investigated.