


Messaging patterns

03/01/2018 • 2 minutes to read • Contributors      [all](#)

The distributed nature of cloud applications requires a messaging infrastructure that connects the components and services, ideally in a loosely coupled manner in order to maximize scalability. Asynchronous messaging is widely used, and provides many benefits, but also brings challenges such as the ordering of messages, poison message management, idempotency, and more.

Pattern	Summary
Claim Check	Split a large message into a claim check and a payload to avoid overwhelming a message bus.
Competing Consumers	Enable multiple concurrent consumers to process messages received on the same messaging channel.
Pipes and Filters	Break down a task that performs complex processing into a series of separate elements that can be reused.
Priority Queue	Prioritize requests sent to services so that requests with a higher priority are received and processed more quickly than those with a lower priority.
Publisher-Subscriber	Enable an application to announce events to multiple interested consumers asynchronously, without coupling the senders to the receivers.
Queue-Based Load Leveling	Use a queue that acts as a buffer between a task and a service that it invokes in order to smooth intermittent heavy loads.
Scheduler Agent Supervisor	Coordinate a set of actions across a distributed set of services and other remote resources.