# Choose a messaging model in Azure to loosely connect your services

Many applications consist of programs that run on several different computers or devices. In such distributed applications, messages must be sent between the components across networks and long distances. Even on the same server or in the same data center, loosely coupled architectures require mechanisms for components to communicate. Reliable messaging is often a critical problem.

You plan to solve these issues by using one or more of the following technologies:

1. Azure Storage queues,
2. Azure Event Hubs,
3. Azure Event Grid,
4. Azure Service Bus.

## Choose whether to use messages or events

### What is a message?

* contains raw data, **produced by one** component, that will be **consumed by another** component.
* contains the **data itself**, not just a reference to that data.
* The sending component **expects** the destination component to **process** the message content **in a certain way**

### What is an event?

* Events are **lighter weight** notification that indicates that something happened
* Events are most **often used for broadcast communications**
* event may be sent to multiple receivers, or to none at all
* Events are often intended to "**fan out**," or have **a large number of** **subscribers** for each publisher
* The publisher of the event has **no expectation** about the action a receiving component takes
* Some events are discrete units and unrelated to other events
* Some events are part of a related and ordered series

### How to choose messages or events

For each communication, consider the following question:

#### Does the sending component expect the communication to be processed in a particular way by the destination component?

# SUMMARY - How to choose a communications technology

## Is the communication a message?

(Does the sending component expect the communication to be processed in a particular way by the destination component?)

If so, consider using Service Bus or Storage Queues.

### **Does a message have more than one destination?**

If so, use a Service Bus Topic.

### Do you need any special features?

### (E.g., Transactional messages, At most/least once guarantee, role-based access, messages size 256 KB (standard tier) or 100 MB (premium tier))

### (Your queue size will NOT grow larger than 80 GB.)

If so, use a Service Bus Queue.

### Are your requirements simple?

### (Audit trail of all messages, Supports unlimited queue size (500 TB),

### target throughput of 2000 messages per second, track progress, code quickly)

(max message size is 64 KB)

If so, use a Storage Queue.

## Is the communication an event?

(Does the sending component have to simply notify the destination component that event has occurred without sending any data?)

If so, consider using Event Grid or Event Hubs.

### Is there high-flow stream of communications?

(It can also be configured to scale dynamically, when required, to handle increased throughput.)

If so, consider using Event Hubs. (Mostly used to create a data point for analytics)

### Is the event a state change in your Azure services?

### (Throughput in the range of millions per second)

Is so, use Event Grid.

# Azure message-based delivery with queues

Suppose you are planning the architecture for your music-sharing application. You want to ensure that music files are uploaded to the web API reliably from the mobile app. You then want to deliver the details about new songs directly to the app when an artist adds new music to their collection. This scenario is a perfect use of a message-based system and Azure offers two solutions to this problem:

1. Azure Queue Storage, 2. Azure Service Bus Queues, 3. Azure Service Bus Topics

## What is Azure Queue Storage?

Queue storage is a service that uses Azure Storage to store large numbers of messages that can be securely accessed from anywhere in the world using a simple REST-based interface. Queues can contain millions of messages, limited only by the capacity of the storage account that owns it.

## What is Azure Service Bus Queues?

Service Bus is a message broker system intended for enterprise applications. These apps often utilize multiple communication protocols, have different data contracts, higher security requirements, and can include both cloud and on-premises services. Service Bus is built on top of a dedicated messaging infrastructure designed for exactly these scenarios.

## What are Azure Service Bus Topics?

Azure Service Bus topics are like queues, but can have multiple subscribers. When a message is sent to a topic instead of a queue, multiple components can be triggered to do their work.

Internally, topics use queues. When you post to a topic, the message is copied and dropped into the queue for each subscription. The queue means that the message copy will stay around to be processed by each subscription branch even if the component processing that subscription is too busy to keep up.

## Benefits of queues

### Increased reliability

Queues increase the reliability of the message exchange because, at times of high demand, messages can wait until a destination component is ready to process them.

### Message delivery guarantees

Queuing systems usually guarantee delivery of each message in the queue to a destination component. However, these guarantees can take different approaches:

1. **At-Least-Once Delivery**
2. **At-Most-Once Delivery**
3. **First-In-First-Out (FIFO)**
4. **Transactional support**

## Choose a service

### Use Service Bus topics if you:

* Need multiple receivers to handle each message

### Use Service Bus queues if you:

* Need an At-Most-Once delivery guarantee.
* Need a FIFO guarantee.
* Need to group messages into transactions.
* Want to receive messages without polling the queue.
* Need to provide a role-based access model to the queues.
* Need to handle messages larger than 64 KB but less than 100 MB. The maximum message size supported by the standard tier is 256 KB and the premium tier is 100 MB.
* Know Queue size will not grow larger than 1 TB. The maximum queue size for the standard tier is 80 GB and for the premium tier, it's 1 TB.
* Want to publish and consume batches of messages.

### Use Queue storage if you:

* Need an audit trail of all messages that pass through the queue.
* Expect the queue to exceed 1 TB in size, it can be up to 500 TB.
* Want to track progress for processing a message inside of the queue.

**Use Storage queues when you want a simple and easy-to-code queue system.**

**For more advanced needs, use Service Bus queues.**

**If you have multiple destinations for a single message, but need queue-like behavior, use Service Bus topics.**

# Choose Azure Event Grid

## What is Azure Event Grid?

* Event Grid distributes events from different sources, such as Azure Blob storage accounts or Azure Media Services, to different handlers, such as Azure Functions or Webhooks.
* Event Grid was created to make it easier to build event-based and serverless applications on Azure.
* Event Grid supports most Azure services as a publisher or subscriber and can be used with third-party services.
* It provides a dynamically scalable, low-cost, messaging system that allows publishers to notify subscribers about a status change.

## Azure Event Grid Components:

* Events: What happened.
* Event sources: Where the event took place.
* Topics: The endpoint where publishers send events.
* Event subscriptions: The endpoint or built-in mechanism to route events, sometimes to multiple handlers. Subscriptions are also used by handlers to filter incoming events intelligently.
* Event handlers: The app or service reacting to the event.

Application, table

Description automatically generated

## What is an event?

Events are the data messages passing through Event Grid that describe what has taken place.

Each event is self-contained, can be up to 64 KB, and contains several pieces of information based on a schema defined by Event Grid:

[ {

// The full resource path to the event source. Event Grid provides this value.

"topic": string,

// Publisher-defined path to the event subject.

"subject": string,

// The unique identifier for event.

"id": string,

// One of the registered event types for this event source. This is a value you can create filters // against, e.g. CustomerCreated, BlobDeleted, HttpRequestReceived, etc.

"eventType": string,

// The time the event was generated based on the provider's UTC time.

"eventTime": string,

// Specific information that is relevant to the type of event.

// However, the actual object that was changed is not part of the event data. Instead, a URL or / // identifier is often passed to reference the changed object.

"data":{

object-unique-to-each-publisher

},

"dataVersion": string,

"metadataVersion": string

} ]

## When should you use Event Grid?

Use Event Grid when you need these features:

* Simplicity: It is straightforward to connect sources to subscribers in Event Grid.
* Advanced filtering: Subscriptions have close control over the events they receive from a topic.
* Fan-out: You can subscribe to an unlimited number of endpoints to the same events and topics.
* Reliability: Event Grid retries event delivery for up to 24 hours for each subscription.
* Pay-per-event: Pay only for the number of events that you transmit.

**Event Grid is a simple but versatile event distribution system. Use it to deliver discrete events to subscribers, which will receive those events reliably and quickly.**

# Choose Azure Event Hubs

There are certain applications that produce a massive number of events from almost as many sources. We often hear the term "Big Data" applied to these situations, and they require unique infrastructure to handle them.

## What are Azure Event Hubs?

Event Hubs is an intermediary for the publish-subscribe communication pattern.

Unlike Event Grid, however, it is optimized for extremely high throughput, a large number of publishers, security, and resiliency.

Whereas Event Grid fits perfectly into the publish-subscribe pattern in that it simply manages subscriptions and routes communications to those subscribers, **Event Hubs performs quite a few** **additional services**.

### Partitions:

As Event Hubs receives communications, it divides them into partitions.

Partitions are buffers into which the communications are saved.

Because of the event buffers, events are not completely ephemeral, and an event isn't missed just because a subscriber is busy or even offline. The subscriber can always use the buffer to "catch up".

### Capture:

Event Hubs can send all your events immediately to Azure Data Lake or Azure Blob storage for inexpensive, permanent persistence.

### Authentication:

All publishers are authenticated and issued a token. This means Event Hubs can accept events from external devices and mobile apps, without worrying that fraudulent data from prankers could ruin our analysis.

## When should you use Event Hubs:

Use Event Hubs when you need these features:

* Support authenticating a large number of publishers.
* Save a stream of events to Data Lake or Blob storage.
* Aggregation or analytics on your event stream.
* Reliable messaging or resiliency.

Event Hubs lets you build a big data pipeline capable of processing millions of events per second with low latency. It can handle data from concurrent sources and route it to a variety of stream-processing infrastructures and analytics services. It enables real-time processing and supports repeated replay of stored raw data.