<https://blog.miraclesoft.com/understanding-azure-service-bus-a-comprehensive-overview/#:~:text=Service%20Bus%20ensures%20reliable%20message,high%20availability%20and%20fault%20tolerance>.

**Types of Azure Service Bus Topic Subscription Filters**

* SQL Filter
* Boolean Filter
* Correlation Filter

<https://turbo360.com/blog/azure-service-bus-topic-filter>

**Options To Create Topic Subscription Rules**

Creation of Service Bus Topic and Subscription can be done in the Azure Portal. Adding rules to the Subscription can be done in the following ways:

* ARM Template
* SDK
* [Third-party application](https://www.serverless360.com/azure-service-bus-monitoring-management)

<https://learn.microsoft.com/en-us/azure/service-bus-messaging/topic-filters>

* **What is Azure Service Bus, and what problems does it solve?** (e.g., decoupling systems, asynchronous communication, message brokering)

Azure Service Bus is a cloud-based, enterprise-level message broker that facilitates reliable and scalable communication between applications and services. It decouples applications, enabling them to exchange messages without needing to be directly connected or online at the same time. This solves problems related to distributed systems, such as: 1) simplifying communication between different applications and services, 2) enabling load balancing and scaling, 3) providing reliable and fault-tolerant message delivery, and 4) supporting complex messaging patterns like publish-subscribe.

Here's a more detailed breakdown:

What it is:

* **Cloud-based messaging service:**

Service Bus is a fully managed service in Azure that handles message queuing and publish-subscribe patterns.

* **Decoupling:**

It decouples applications, meaning they don't need to be aware of each other's specific details or availability to communicate.

* **Reliable and Scalable:**

Service Bus ensures messages are delivered reliably and can handle high volumes of traffic, scaling automatically to meet demand.

* **Supports various patterns:**

It offers features like message queues, topic and subscription models, and message sessions to support different architectural patterns.

Problems it solves:

* **Complexity of distributed systems:**

Service Bus simplifies the integration of diverse applications and services, especially in cloud environments, by acting as a central messaging hub.

* **Reliability and fault tolerance:**

It ensures messages are delivered even if the sender or receiver is temporarily unavailable, preventing message loss.

* **Load balancing and scaling:**

By distributing messages across multiple consumers, Service Bus helps manage traffic spikes and ensures applications can handle increased load.

* **Tight coupling and dependencies:**

It reduces dependencies between applications, making them more resilient to changes and failures.

* **Complex messaging workflows:**

Features like publish-subscribe and message sessions enable building sophisticated messaging patterns for coordinating complex tasks and workflows.

**Reliable Messaging**

Service Bus ensures reliable message delivery by providing features such as message durability, at-least-once delivery semantics, and message retry policies. Messages are persisted and replicated across multiple nodes within the Service Bus namespace for high availability and fault tolerance.

* [Asynchronous messaging patterns and high availability](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-async-messaging)
* [Azure Service Bus Geo-disaster recovery](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-geo-dr)
* [Handling outages and disasters](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-outages-disasters)
* [Availability zones](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-outages-disasters#availability-zones)
* [Geo-Disaster Recovery](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-geo-dr) (paired namespace).
* [Geo-Replication](https://learn.microsoft.com/en-us/azure/service-bus-messaging/service-bus-geo-replication) (full data replication across regions).

**Difference Between Service Bus Topics and Queues**

**Service Bus Queues**

* **One-to-One Communication:** Service Bus Queues are designed for one-to-one communication. A message sent to a queue is received and processed by a single receiver.
* **Message Delivery:** Messages are stored in the queue until they are retrieved and processed by the receiving application.
* **Load Balancing:** Queues can be used for load balancing. Multiple receivers can pull messages from the same queue, but each message is delivered to only one receiver, ensuring a distributed workload.
* **FIFO Ordering:** Queues generally follow a First-In-First-Out (FIFO) order for message delivery, ensuring that messages are processed in the order they were sent.

**Service Bus Topics**

* **One-to-Many Communication:** Service Bus Topics are designed for one-to-many communication. A message sent to a topic can be received by multiple subscribers.
* **Subscriptions:** Topics support multiple subscriptions, each of which can receive a copy of the message. This allows different parts of an application to independently process the same message.
* **Filtering and Rules:** Subscriptions can have filters and rules applied to them, allowing each subscriber to receive only the messages they are interested in.
* **Scalability:** Topics are useful for scenarios where messages need to be broadcast to multiple receivers, making it easy to scale out message processing across different services or components.

**Use Cases**

**1. Decoupled Microservices Communication:**Service Bus facilitates communication between microservices in a distributed system, allowing them to exchange messages asynchronously and decoupling them from each other.

**2. Event-Driven Architecture:**Service Bus is commonly used in event-driven architectures for event publishing, subscription, and processing. It enables applications to publish events and notify interested subscribers asynchronously.

**3. Integration Scenarios:**Service Bus is used for integrating applications and services across different environments, including on-premises systems, cloud applications, and hybrid environments. It serves as a reliable message broker for data integration and workflow orchestration.

**4. Queue-Based Processing:**Service Bus queues are used for implementing queue-based processing patterns, such as task distribution, workload leveling, and asynchronous job processing.

**5. Scheduled and Deferred Messages:**Service Bus supports scheduling messages for future delivery and deferring message processing until a specified time, enabling scenarios such as delayed notifications, reminders, and scheduled tasks.

Azure Service Bus is a cloud-based message broker that enables asynchronous communication between applications and services. It is used to decouple applications, route data, and coordinate transactional work. Common use cases include e-commerce order processing, financial transactions, telecommunications, and data pipelines.

Here's a more detailed look at the use cases:

1. E-commerce Platforms:

* **Order Processing:** Service Bus can manage the flow of order information between different services like storefronts, payment gateways, and inventory management systems.
* **Payment Notifications:** Ensures reliable delivery of payment status updates to relevant services.
* **Inventory Management:** Facilitates the update of inventory levels based on sales and other transactions.

2. Financial Services:

* **Secure Messaging:** Provides a secure and reliable way to communicate between banking systems, including ATMs, mobile banking, and payment processors.
* **Transaction Processing:** Enables real-time transaction processing while maintaining data integrity.

3. Telecommunications:

* **Event-driven Notifications:** Handles event-driven notifications for various services like service provisioning and billing updates.
* **Service Provisioning:** Manages the automated provisioning of new services and resources.
* **Billing Updates:** Ensures timely and accurate billing updates across different systems.

4. Data and Event Pipelines:

* **Real-time Analytics:**

Azure Service Bus can be used to build real-time data pipelines for analytics and monitoring across different systems.

* **Big Data Processing:**

Facilitates the processing and analysis of large volumes of data by routing it to different processing components.

5. Integration Scenarios:

* **Decoupling Applications:**

Service Bus decouples applications, allowing them to communicate asynchronously and independently.

* **Integration with Logic Apps:**

Provides easy integration with Azure Logic Apps for building complex workflows.

6. Other Use Cases:

* **Cloud OCR Capability:** Can be used to handle the asynchronous processing of scanned documents and images.
* **Provisioning Azure Resources:** Used to automate the provisioning of Azure resources.
* **Serverless Home Automation:** Can be used to build serverless home automation solutions that respond to events and control devices.

Key Benefits of using Azure Service Bus:

* **Reliability:**

Provides guaranteed message delivery and ensures messages are processed once and only once.

* **Scalability:**

Can handle large volumes of messages and scale to meet the demands of various workloads.

* **Decoupling:**

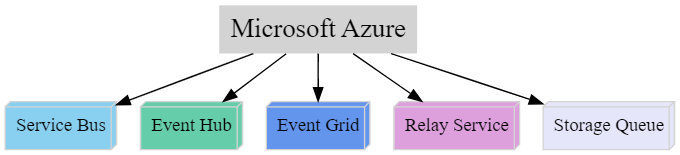
Enables applications to communicate without being tightly coupled, increasing flexibility and maintainability.

* **Security:**

Provides robust security features for message protection and access control.

* **Advanced Features:**

Offers features like duplicate detection, message sessions, and scheduled messages.



**1. Azure Service Bus**

* Provides **durable brokered messaging**, meaning messages are stored securely while in transit.
* Supports **point-to-point messaging with queues and publish/subscribe messaging with topics and subscriptions**.
* Offers advanced features like **duplicate detection, sessions, scheduled messages, and message expiration**.
* Example:  
  - Scenario: An e-commerce platform where different services (like order processing, inventory management, and shipping) need to communicate reliably.  
  - Usage: Service Bus queues and topics can ensure that orders are processed in sequence and messages are not lost, even if a service goes down temporarily.

**Azure Message Bus Features**

1. **Point-to-Point and Publish/Subscribe Messaging:**You can**send messages directly from one application to another** (point-to-point) or **broadcast messages to multiple applications** (publish/subscribe), providing flexibility in how messages are routed.
2. **Dead Lettering: Messages that can’t be processed successfully** are moved to a dead letter queue, where they can be reviewed and fixed later.
3. **Message Sessions:** If you have **related messages** that need to be **processed together**, you can **group them into sessions** so they are handled by the same application.
4. **Request/Response Pattern:** This allows you to match responses with the original requests, **enabling two-way communication**even in an asynchronous system.
5. **Message Deferral:** Sometimes, an application may need to **delay processing a message.**Deferral allows the message to stay in the queue and be processed later.
6. **Scheduled Messages:** You can set messages to be **sent at a specific time in the future.**
7. **Duplicate Detection:** To **prevent duplicate messages from being processed multiple times**, the system can detect and ignore duplicates.
8. **Time to Live (TTL):** **Messages can be set to expire after a certain period**. Expired messages won’t be processed and **can be moved to the dead letter queue** if needed.

**Protocols**

* **AMQP (Advanced Message Queuing Protocol):** A widely used, open messaging protocol that’s the default for the Azure Service Bus SDK and is compatible with many applications.
* **HTTP:** Useful for environments with strict firewall rules and offers good cross-platform compatibility.

<https://medium.com/@darshana-edirisinghe/azure-service-bus-part-1-ensuring-reliable-messaging-in-your-cloud-architecture-ce2df8ae2015>