# Publish Subscribe Architecture

Group name: Exploding Kitten

## What is publish subscribe architecture?

Publish subscribe architecture is a message protocol where sender of the messages are called publisher. The publishers do not specify directly to whom the message to be sent which is to the subscribers. Event bus acts as a intermediary that collects all messages from the publisher and forward to subscribers based on their interest.
Publishers send messages without any knowledge of the subscribers and subscribers receive messages without any knowledge of the publishers. Messages could be filtered based on topic, content or type,etc by different subscription models.

## Have its own vocabulary for its components and connectors?

Components:

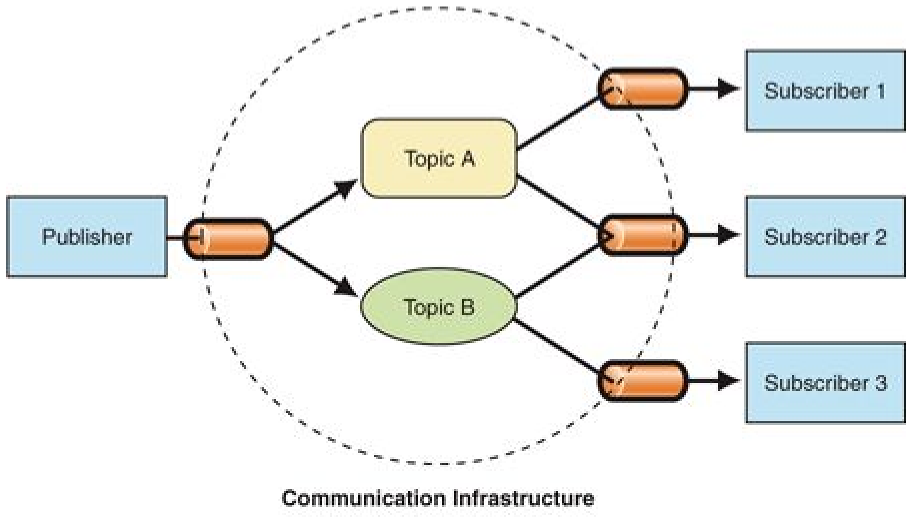
- Publishers:
    - Senders of the messages
- Subscribers:
    - Receivers of the messages

Connectors:

- Event buses:
    - Collect all messages from the publishers
    - Forward all messages to subscribers based on their interest

## Impose specific topological constraints?

Publishers and subscribers can not talk to each other directly.



(Reference from: https://msdn.microsoft.com/en-us/library/ff649664.aspx)

## Most applicable to specific kinds of problems?

1. Application sending different types of messages
2. Application consuming different types of messages
3. Senders can send messages without knowledge about the receivers
4. Receivers get messages without knowledge about the senders
5. Need to add/remove consumer frequently
6. Application that infrequently updated

## Advantages:

### Loose coupling:

Publishers are not coupled to subscribers, and do not know the identity and numbers of the subscribers. Unlike traditional client-server paradigm, the client cannot send messages to the server while the server process is off, nor can the server receive messages when the client is not running. Publishers are only responsible for posting message an intermediary message broker.

### Scalability:

The pub/sub system can support large number of client and data transfer as result of loose coupling. Publishers just publish all the required messages to the broker instead of directly talking to the subscribers, and subscribers can book the messages that only meet their interest.

### Improved testability:

Because of classifying the messages to several topics, the number of messages which are required for testing are decreased. Testers can easily check if the messages are sent to the topic that are not responds to their subscriptions.

## Disadvantages:

### Unstable delivery:

In a pub/sub system, a publisher cannot assume a subscriber is listening when publisher post a message. For example, if a logger used to subscribe the messages of problem of error, when it crash, the problem publisher may not aware that the logger has failed, and the error message will not be displayed.

### Inflexibility of data:

It is difficult to modify the data structure in a pub/sub system because all subscribers must change their structure in order to accept new format. It will be very difficult when subscribers are external.

### Bottleneck:

With the growth in the scale of a pub/sub system, the message broker may become a bottleneck for data transition. Because it is required not only receive messages from publisher, but also send messages to subscribers. Therefore, with high intensity data flow, the pub/sub system may be slow.

## Support/inhibit specific NFPs?

- Low complexity
  - Subscribers can receive messages concurrently.
  - Communication between publishers and subscribers done by event bus.
  - Subscriber can only receive messages which they interested in.
  - Publisher and subscriber do not need to know each other
  - Publisher and subscriber can work asynchronously
- Evolvability
  - Components can evolve independently without changing the interface
  - Subscribers can be added and removed easily

## Example

### Description:

This is a simple example demonstrate publisher subscriber architecture. We consider the whole network is a dining hall. Chef is a publisher, waiter is an event bus and the customers are subscribers.

### Scenarios:

1. Chef cooks chicken, beef and vegetable salad and give them all to the waiter. The waiter give chicken to Bob who like chicken, give beef to Alex who like beef and give vegetable salad to Leo who like vegetable salad. (Common case)
2. Chef cooks chicken beef and vegetable salad as usual in the morning and then he go to see movies in the afternoon. Bob, Alex, and Leo come in the afternoon can still enjoy their favourite meals. (Advantage 1: Loose coupling)
3. Chef decides to cook fish for customers and many new customers come to the dining hall. The dining hall still works fine. (Advantage 2: Scalability)
4. Because of cold weather, only a few customer come to the dining hall but chef still cook a lot of dishes like usual. Therefore, the service table cannot hold to many dishes and some of them just be dropped off and wasted. (Disadvantage 1: Unstable delivery)
5. The dining hall decide to change their accepted payment methods. It used to accept debit and visa, but it want to only accept cash now. The waiter noticed all yesterday's customer that they should prepare cash from today. However, there are still some customer who did not come yesterday but come today cannot buy their food because they did not know they should prepare cash. (Disadvantage 2: Inflexibility of data)