

Publish–Subscribe

PUBLISH-SUBSCRIBE VS. EVENT-BASED

There are two important differences that differentiate the publish-subscribe architectural style from the event-based architectural style.

First, unlike the event-based architectural style, the components are either publishers or subscribers.

Publishers send messages while subscribers can subscribe or unsubscribe to receive these messages. A component cannot be both a publisher and a subscriber.

And second, the relationship between the publisher and the subscribers may vary in form and closeness.

SIMILARITIES TO EVENT-BASED

Like the event-based architectural style, the publish-subscribe architectural style relies on implicit invocation. Publishers and subscribers are not explicitly connected, but instead, there is a loose coupling between them.

IMPLEMENTATION

In a simple implementation of the publish-subscribe architectural style; to receive messages, a subscriber registers their interest to publishers through a call-back.

Publishers maintain a list of subscribers and communicate to them through procedure calls to the registered call-backs. In this way, subscribers are informed of messages implicitly.

This implementation is well suited to applications where each publisher has a relatively small number of subscribers. Every time a new message is ready to be published, the publisher makes a procedure call to each subscriber to deliver the new message. This can become a lot of work for the publisher as the number of subscribers grows.

LARGE SCALE IMPLEMENTATIONS

Alternatively, the publish-subscribe architectural style can be adapted to deal with large scale applications, where a single publisher may have many subscribers, by designing a system with more loosely coupled components.

This is achieved by introducing a connector to distribute and possibly filter the messages to the subscribers. The connector could be an event bus or a network protocol.

The connector creates more distance between publishers and subscribers and **relieves work for the publisher**. These connectors may provide **complex filtering of messages** before they are delivered to the subscribers.

If an event bus is used as a connector, the system would be similar to the event-based architectural style, in that subscribers register with the event bus to receive messages from a particular publisher, and the publisher sends messages to the event bus for delivery to its subscribers.

Because the event bus mediates communication, **publishers may not be aware of the subscribers**. However, remember there is an important difference here in the publish-subscribe architectural style, **publishers can only send messages, not receive them, and subscribers can only receive messages, not send them**.

CONCLUSION

The flexible loose coupling between publishers and subscribers makes publish-subscribe systems scalable and flexible.

While designing your system, it's important to consider that using this architectural style may make it **impossible to modify a published message**. Since each subscriber has their own copy of the message, a correction message may be the only way to convey a change.

Some example systems that can manifest this style are **mailing lists, RSS feeds, and mobile messaging subscriptions**.