

# BDI Dynamic event notification distribution on static networks using webhooks



# Colophon

---

## BDI Dynamic event notification distribution on static networks using webhooks

### Authors

Ewout Bouwman  
Herman Wagter

January 2025

© Connekt



# Summary

---

The reference architecture of the BDI framework introduces the concept of the choreography of the distribution of (notifications of) events.

The assumption in the framework is that an event broker (pub/sub) is used to create topic or channels between parties, provision the temporary network of parties involved.

A professional event broker like Kafka or Pulsar can be oversized and complex for smaller business networks and smaller entities. It may well be possible that SaaS offerings arise to service this part of the market.

Yet the demand has been visible for a more lightweight option. The option to use webhooks<sup>1</sup>, or 'user-defined HTTP callbacks' has been proposed.

It is possible to create an event choreography in a static network using webhooks between parties as an event communication protocol.

The choreography is created by a framework that describes agreed upon identifications, lists and procedures. Such a framework is less powerful and automated than a full fledged event-broker, but can be useful in practice.

Adding more layers of subcontracting requires a more extensive investigation of the required procedures



---

<sup>1</sup> [Webhook - Wikipedia](#)

# Contents

---

<b>1.</b>	<b>Introduction</b>	<b>5</b>
<b>2.</b>	<b>Webhooks vs Pub/Sub</b>	<b>7</b>
<b>3.</b>	<b>Network connections and business relationships</b>	<b>8</b>
	3.1 Network connections follow business relationships	8
	3.2 Volatile projects, orders and relationships	8
<b>4.</b>	<b>Event Choreography</b>	<b>9</b>
	4.1 Initiation	9
	4.2 Notifications sent by subcontractor	9
	4.3 Notifications received by subcontractor	11
	4.4 Closure	11
<b>5.</b>	<b>Conclusion</b>	<b>12</b>

# 1 Introduction

---

The reference architecture of the BDI framework introduces the concept of the choreography of the distribution of (notifications of) events.

## Event Choreography | Public

### Choreography of event distribution

In supply chains the chain of business activities starts when a Seller and Buyer agree upon the transaction. This agreement typically includes terms related to transport, insurance, customs, the handover of responsibilities, and payments. The successful execution of this agreement often requires coordination among a large set of actors, including authorities and their subcontractors. This coordination is managed through a 'choreography' of actions, where each action is triggered by planned or executed events.

### Subcontractors and principals

In this context, the Seller and Buyer serve as the Principals, each responsible for their part of the agreement. Typically, each Principal selects preferred contractors to fulfil their portion of the agreement. These main contractors, in turn, become Principals to their own subcontractors, and this chain continues down the line.

The key challenge here is distributing notifications within this dynamic and temporary data exchange network, to ensure effective and efficient coordination of activities, without overburdening the network with traffic.

### Provisioning an instance

The BDI framework assumes that commercial relationships between Principals and Subcontractors are established before any actual orders are placed. In this setup, URLs are known, and through a discovery mechanism, the URIs of endpoints for each party are also known. Digital identity and trust are established within the respective associations of each party.

Each Principal is responsible for provisioning a temporary network of subcontractors associated with a specific order.

---

## Channels

The assumption in the framework is that an event broker (pub/sub) is used to create topic or channels between parties, provision the temporary network of parties involved.

Principals and their subcontractors communicate through channels (a.k.a. topics in a pub/sub setting). There are two kinds of channels:

- **Subcontractor specific**

Often a principal works for different orders with a limited set of partners (preferred suppliers). There is a private channel between the principal and each partner. These channels are bidirectional (both the principal and the subcontractor can post notifications). A subcontractor specific channel exists as long as the relation between principal and subcontractor exist and hence may exceed the lifetime of a specific order.

- **Order specific**

A principal creates a channel specific to the execution of a particular order. Such a channel is unidirectional: the principal is the only party who is allowed to post notifications, all involved subcontractors subscribe to this channel and are therefore notified of each message sent by the principal. Order specific channels exist for the duration of the order only.

## Lightweight option

A professional event broker like Kafka or Pulsar can be oversized and complex for smaller business networks and smaller entities. It may well be possible that SaaS offerings arise to service this part of the market.

There is demand for a more lightweight option. The option to use webhooks, or 'user-defined HTTP callbacks' method. It is a simple and efficient for **one** system to notify another about an event.

A webhook defines a static asynchronous path between source (webhook client) and destination (webhook server with URL). When an event occurs at the source site, the client makes a HTTPS POST request to the URL configured for the webhook. The POST request can carry the (notification of an) event, authentication can be based on OAuth.

The question is how to emulate a pub-sub event broker with dynamic topics through static webhooks, configured and provisioned once.

## 2 Webhooks vs Pub/Sub

---

Webhooks and Pub/Sub are two distinct methods for managing message transmission in event-driven systems. While both facilitate communication, their design and use cases make them optimal for different scenarios.

### Control Over Message Delivery

- **Webhooks**  
The sender decides exactly where to deliver the message by specifying the webhook endpoint.
- **Pub/Sub**  
The sender publishes messages to a topic, and subscribers independently choose to receive messages. The sender, in basis, has no control<sup>2</sup> over who subscribes.

### Connectivity

- **Webhooks**  
Require the sender to maintain knowledge of and connectivity to specific endpoints.
- **Pub/Sub**  
Decouples senders from receivers, allowing for scalability and loose coupling in message handling.

---

<sup>2</sup> Pulsar native provides a comprehensive set of security features, including authentication, authorization, encryption, and role-based access control allowing control of message delivery

## 3 Network connections and business relationships

### 3.1 Network connections follow business relationships

The assumption is that all parties involved deploy a 'BDI Connector' with:

- A webhook server with a public URL (DNS discoverable if necessary) than can receive POST requests from (known and authenticated) parties.
  - Authentication can be supported by an association register.
- A webhook client that can send POST requests to the webhook URL's of other parties.

A party can be:

- A 'Principal' giving out contracts to 'Subcontractors'!
- A 'Subcontractor' fulfilling a part of a supply chain.
- Or both at the same time (sub-subcontracting parts of the job).

The assumption is that business relationships are arranged and provisioned in IT-systems by humans before actual orders are placed. During the provisioning of the business relationship the URL of the webhook of party can be registered.

- A Principal registers the URL of the webhook of the subcontractor.
- A Subcontractor registers the URL of the webhook of the Principal.

These relationships are slow-moving.

### 3.2 Volatile projects, orders and relationships

Operational business relationships are project/order based and volatile. Market dynamics ask for agility, flexibility, and responsiveness. The operational logistics therefor involves numerous actors that organize flows of goods and information is shared.

A Principal may issue coherent orders (part of a Project) to a subset of its subcontractors. A Principal may have multiple Projects as work-in-progress in hand. Each Project may use a different subset of the subcontractors known to the Principal. The subset of subcontractors per Project may be modified 'on-the-fly'.

The list of Projects and associated orders connected to subcontractors needs to be maintained by a Principal. The assumption is that the Principal issues unique ID's of project plus order.

A Subcontractor may have multiple concurrent orders as work-in-progress, coming from multiple Principals.

The list of its Principals (customers) and associated orders/projects per Principal needs to be maintained by the subcontractor, including the ID issued by the principals per order/project.

The lists maintained by each principal and each subcontractor are the basis for emulating a topic based event broker.

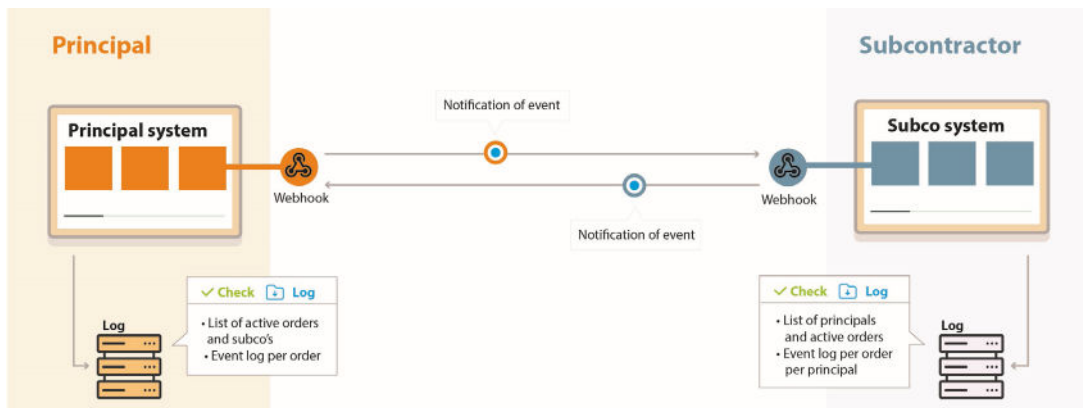
## 4 Event Choreography

### 4.1 Initiation

The assumption is that a Principal initiates a Project by requesting work to be done ('order') to each of a subset of subcontractors. The request is identified as:

- Sent by a Principal to a particular subcontractor.
- ID of project.
- ID of order.

This uniquely identifies the combination and is used from this moment on in exchanges related to this Project.

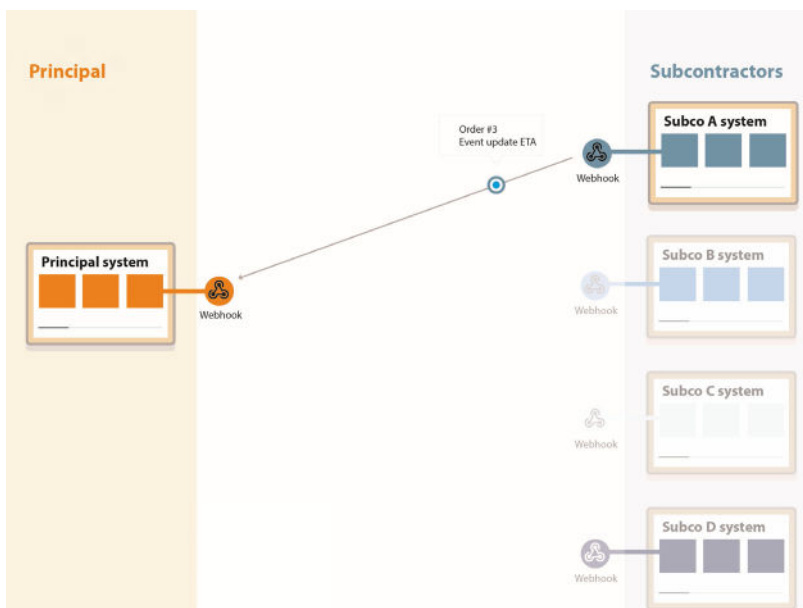


### 4.2 Notifications sent by subcontractor

When after acceptance a particular subcontractor needs to send a notification of an event to the Principal, the notification is pushed to the webhook of the Principal. The notification includes the above identification: order and project. The principal receives through the webhook continuous mixed notifications from all subcontractors for all projects and needs to sort the input to:

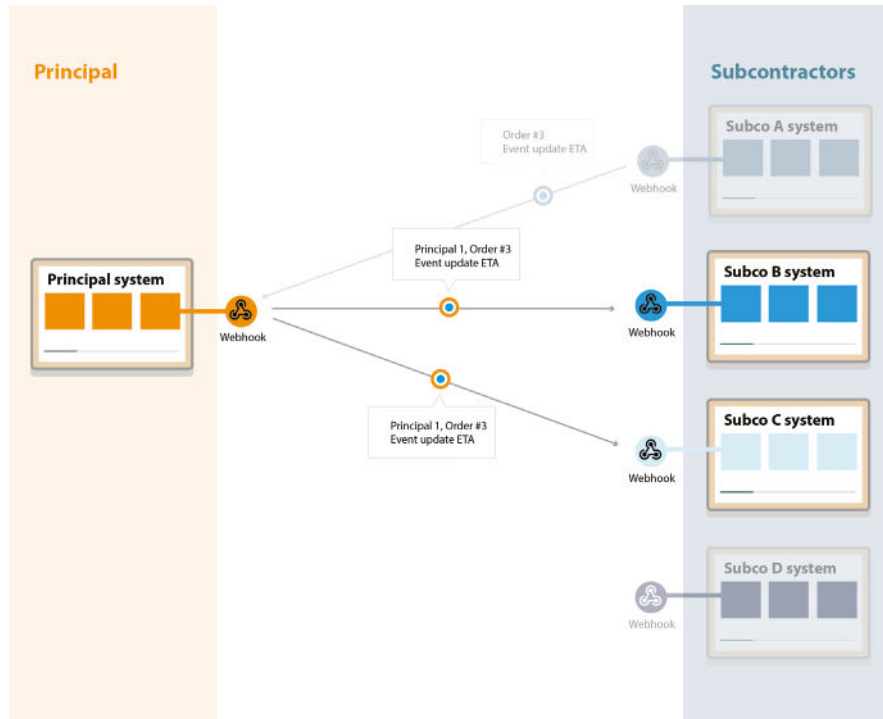
- Project.
- Subcontractor.
- Order.

The input is logged per combination.

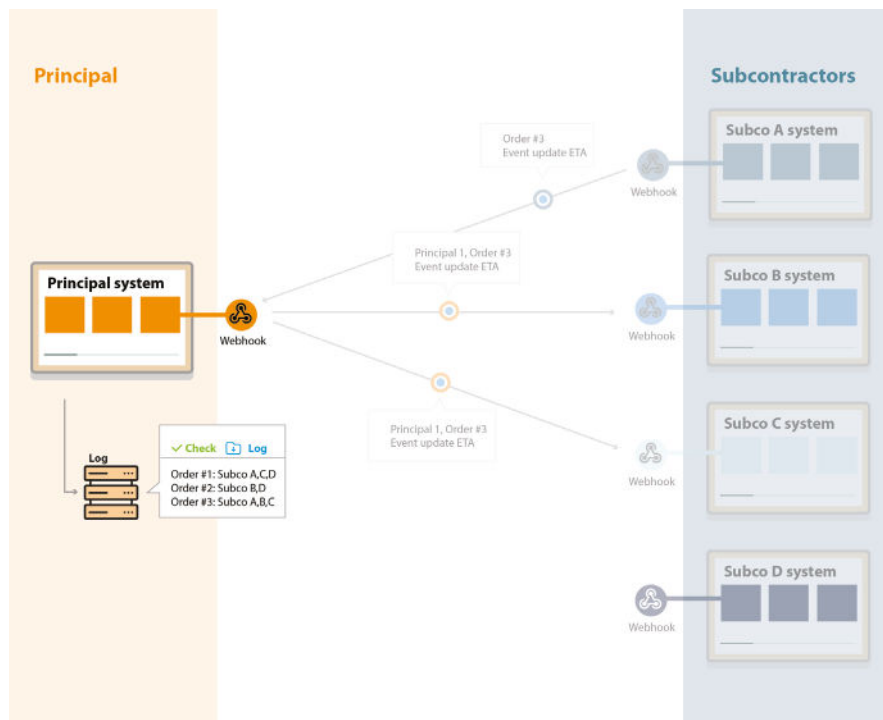


After sorting and evaluating the content of a particular notification the Principal may decide to inform all subcontractors associated with the Project:

- The Principal forwards the notification to all subcontractors in the list connected to the Project.



In this manner an Order Log is created de-facto at each subcontractor.



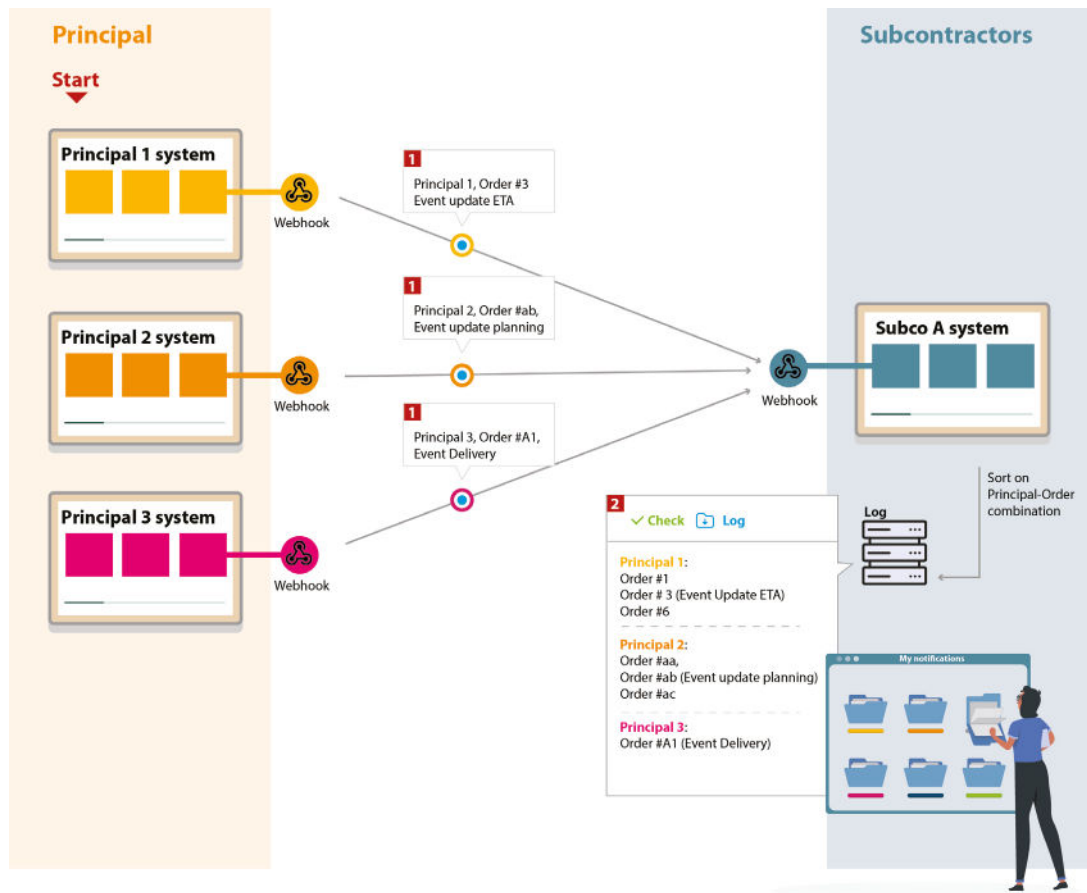
### 4.3 Notifications received by subcontractor

A subcontractor receives through the webhook continuous mixed notifications from all Principals for all projects and needs to sort the input to:

- Principal.
- Project.
- Order.

The input is logged per combination, creating an Order Log.

After sorting and evaluating the content of a particular notification the subcontractor can decide to take action.



### 4.4 Closure

When an order is finished a subcontractor can archive the Order log. The same applies to the Principal for a Project.

These logs can be used for audits and disputes later on.

## 5 Conclusion

---

It is possible to create an event choreography in a static network using webhooks between parties as an event communication protocol.

The choreography is created by a framework that describes agreed upon identifications, lists and procedures. Such a framework is less powerful and automated than a full fledged event-broker, but can be useful in practice.

Adding more layers of subcontracting requires a more extensive investigation of the required procedures.



**BDI, Topsector Logistiek & DIL**

Ezelsveldlaan 59 | 2611 RV Delft | +31 15 251 65 65

[www.bdinetwork.org](http://www.bdinetwork.org) | [www.topsectorlogistiek.nl](http://www.topsectorlogistiek.nl) | [www.datainlogistics.org](http://www.datainlogistics.org)

