## **Coordinating sagas**

A saga’s implementation consists of logic that coordinates the steps of the saga. When a saga is initiated by system command, the coordination logic must select and tell the first saga participant to execute a local transaction. Once that transaction completes, the saga’s sequencing coordination selects and invokes the next saga participant. This process continues until the saga has executed all the steps. If any local transaction fails, the saga must execute the compensating transactions in reverse order. There are a couple of different ways to structure a saga’s coordination logic:

**Choreography**—Distribute the decision making and sequencing among the saga participants. They primarily communicate by exchanging events.

**Orchestration**—Centralize a saga’s coordination logic in a saga orchestrator class. A saga orchestrator sends command messages to saga participants telling them which operations to perform.

**Choreography-based sagas**

One way you can implement a saga is by using choreography. When using choreography, there’s no central coordinator telling the saga participants what to do. Instead, the saga participants subscribe to each other’s events and respond accordingly.

IMPLEMENTING THE CREATE ORDER SAGA USING CHOREOGRAPHY

The participants communicate by exchanging events. Each participant, starting with the Order Service, updates its database and publishes an event that triggers the next participant.

Diagram

Description automatically generated

A picture containing table

Description automatically generated

The Create Order Saga must also handle the scenario where a saga participant rejects the  Order and publishes some kind of failure event. For example, the authorization of the consumer’s credit card might fail. The saga must execute the compensating transactions to undo what’s already been done.

Flow of events when the AccountingService can’t authorize the consumer’s credit card.

The sequence of events in the Create Order Saga when the authorization of the consumer’s credit card fails. Accounting Service publishes the Credit Card Authorization Failed event, which causes  Kitchen Service to reject the Ticket, and Order Service to reject the Order.

Diagram

Description automatically generated

The sequence of events is as follows:

A picture containing table

Description automatically generated

As you can see, the participants of choreography-based sagas interact using publish/subscribe. Let’s take a closer look at some issues you’ll need to consider when implementing publish/subscribe-based communication for your sagas.

**RELIABLE EVENT-BASED COMMUNICATION**

The first issue is ensuring that a saga participant updates its database and publishes an event as part of a database transaction. Each step of a choreography-based saga updates the database and publishes an event. For example, in the *Create Order Saga*, *Kitchen* *Service* receives a *Consumer Verified event*, creates a *Ticket*, and publishes a *Ticket Created event*. It’s essential that the database update and the publishing of the event happen atomically. Consequently, to communicate reliably, the saga participants must use transactional messaging.

The second issue you need to consider is ensuring that a saga participant must be able to map each event that it receives to its own data. For example, when Order Service receives a Credit Card Authorized event, it must be able to look up the corresponding Order. The solution is for a saga participant to publish events containing a correlation id, which is data that enables other participants to perform the mapping.

For example, the participants of the Create Order Saga can use the orderId as a correlation ID that’s passed from one participant to the next. Accounting Service publishes a Credit Card Authorized event containing the orderId from the TicketCreated event. When Order Service receives a Credit Card Authorized event, it uses the orderId to retrieve the corresponding Order. Similarly, Kitchen Service uses the orderId from that event to retrieve the corresponding Ticket.

BENEFITS AND DRAWBACKS OF CHOREOGRAPHY-BASED SAGAS

Graphical user interface, text, application, email

Description automatically generated

And there are some drawbacks:

Table

Description automatically generated with medium confidence