Messaging with NServiceBus

Roland Guijt
MVP | CONSULTANT | TRAINER

@rolandguijt rolandguijt.com

Overview

What is NServiceBus?

How to prepare

Messages

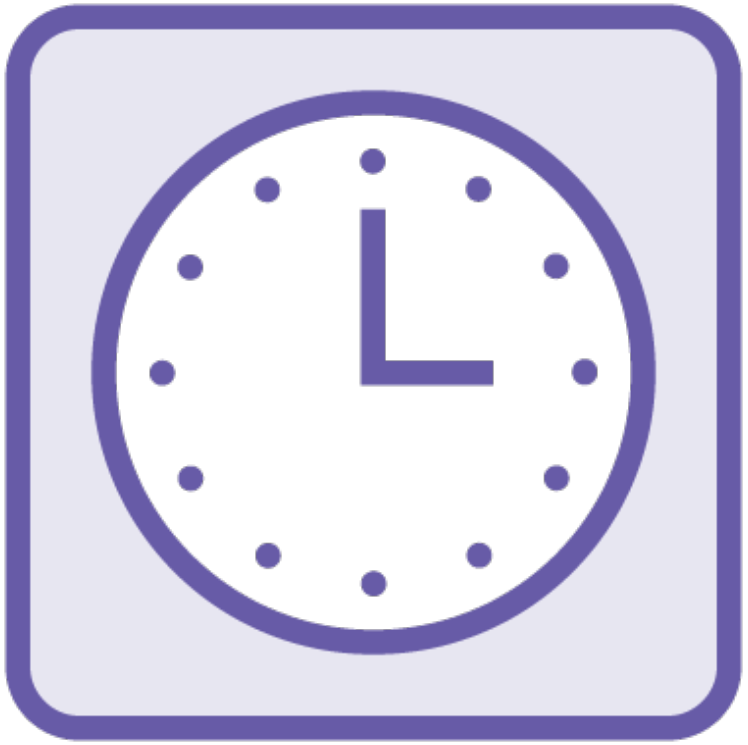
Routing

Configuration

Fault Tolerance

Request/Response

History



Lack of MSMQ support in .NET

Open source since 2007

Udi Dahan - creator

Requires license fee since 2010

What Is NServiceBus?

Framework enabling communication between applications using messaging

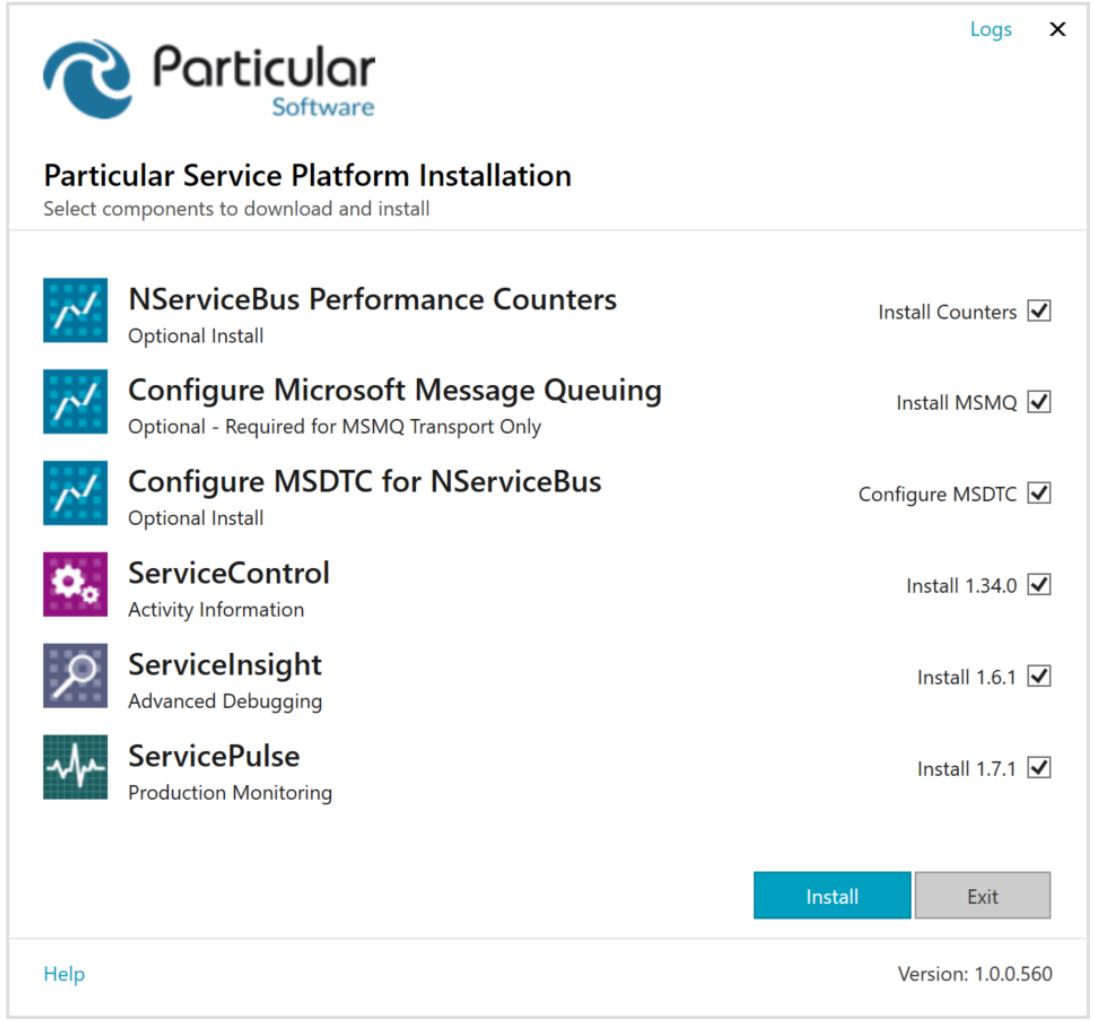
Part of Particular Service Platform

Lies on top of messaging back-ends called transports

Transports are configurable

Open source, but not free

Preparation



http://particular.net/downloads

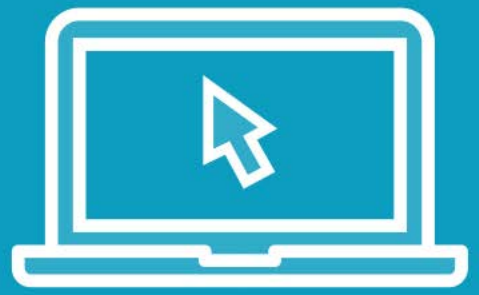
Main NuGet Packages

NServiceBus

NServiceBus.Host

**NServiceBus.
Testing**

Demo



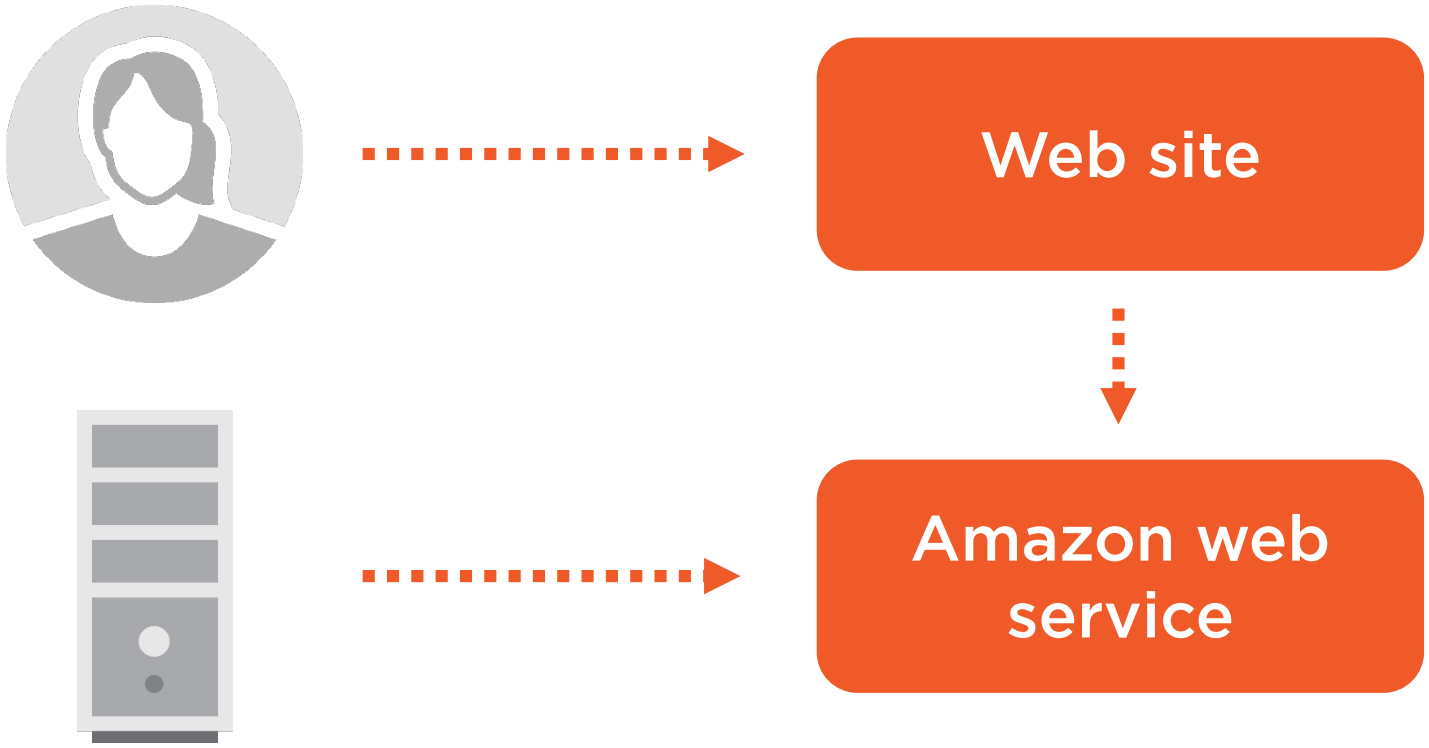
Fire On Wheels business is even better

Orders are getting lost

Too much load on the service

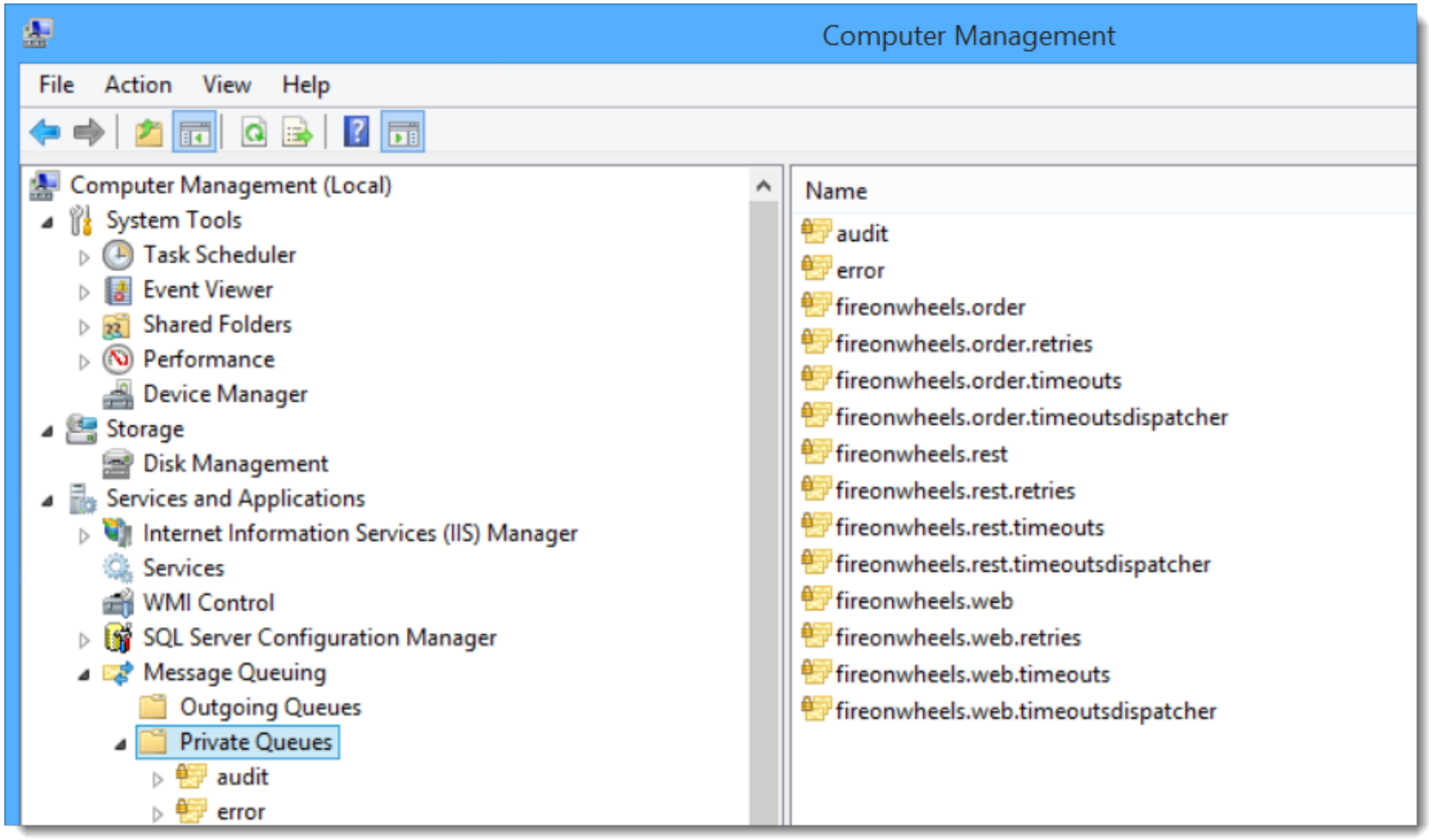
Process orders one by one

The Old Architecture



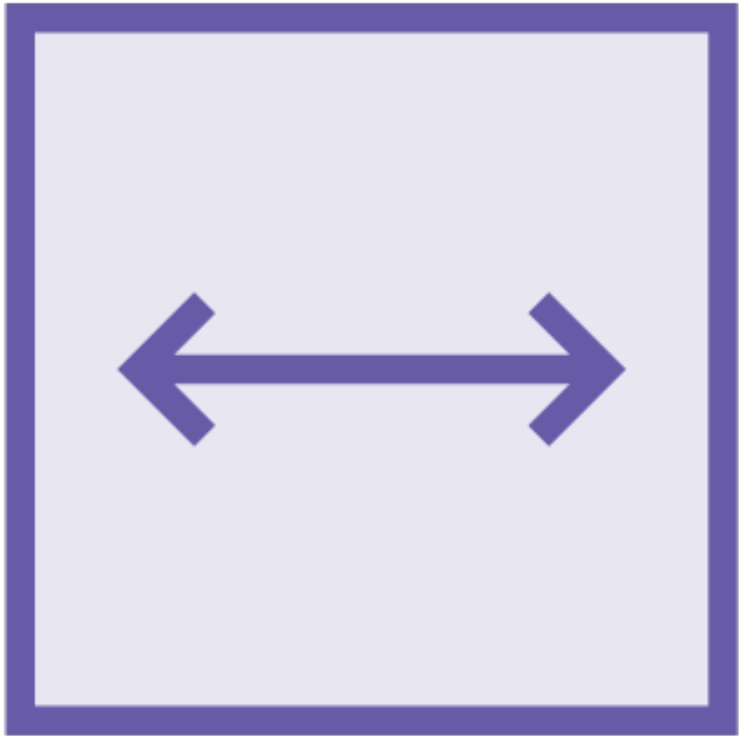
The New Architecture

MSMQ Queues



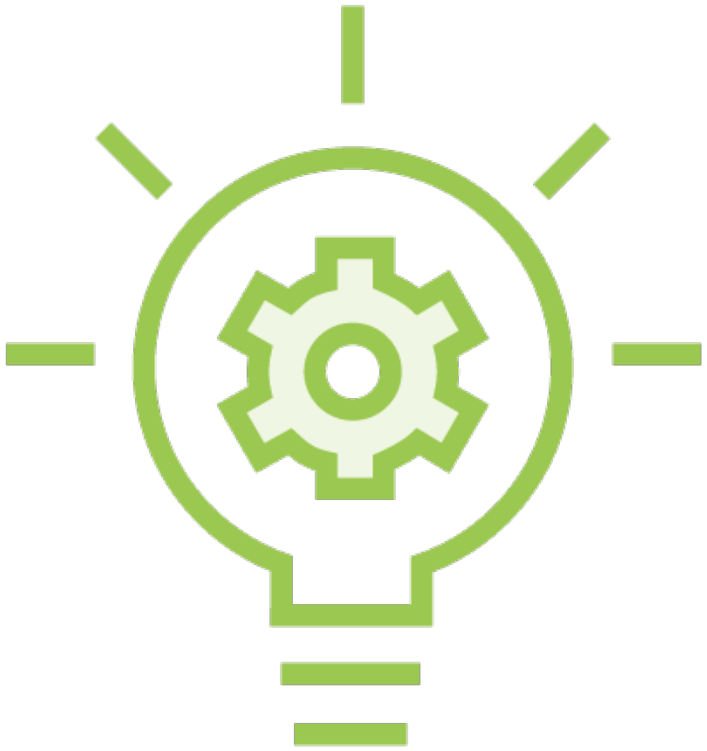
Computer Management -> Service and Applications -> Message Queueing -> Private Queues

Commands



- Messages
- One or more senders
- One receiver
- Send method
- Marked with ICommand
- Imperative: ProcessOrderCommand

Dependency Injection



NServiceBus relies on it

Built into the core

Works with NServiceBus managed types

Or use your own DI container

Assembly Scanning

IHandleMessages

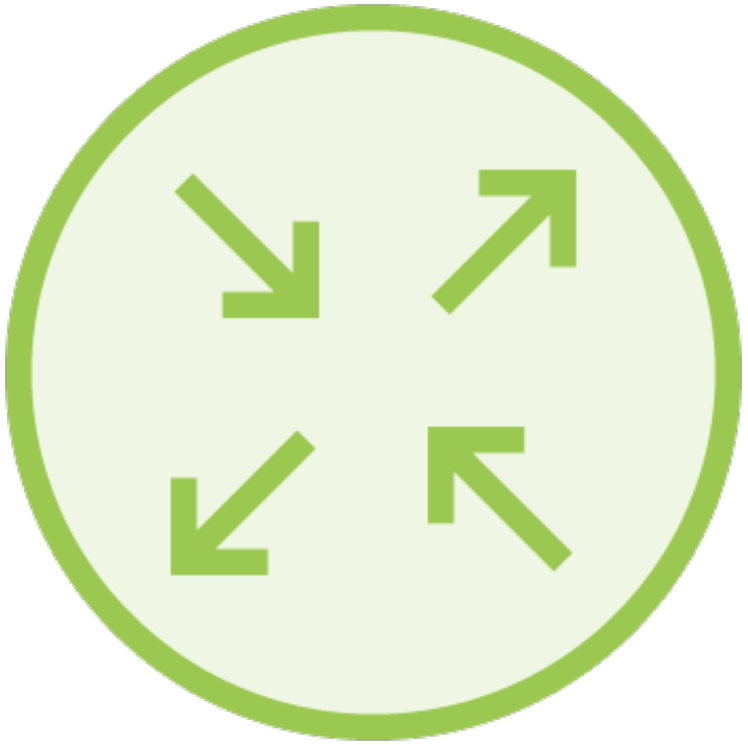
Scans all assemblies

Can be limited

Events

- Messages
- One sender
- One or more receivers
- Publish method
- Publish/Subscribe pattern
- Marked with IEvent
- Past tense: OrderProcessedEvent

Routing



Can be done in config file

No need to redeploy when endpoint names change, etc.

Recommended

Commands: In sender, routed to endpoints that receive the command

Events: In receiver, routed to endpoints that receive the subscription request

```
<UnicastBusConfig>
    <MessageEndpointMappings>
    </MessageEndpointMappings>
</UnicastBusConfig>
```

Routing: The app.config Section

Routing rules or mappings go here

Every mapping in an <add> node

```
<add Messages="assembly" Endpoint="destination" />

<add Assembly="assembly" Type="namespace.type"
Endpoint="destination" />

<add Assembly="assembly" Namespace="MyMessages.Other"
Endpoint="destination" />
```

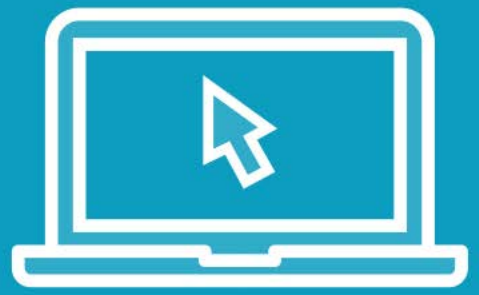
Routing: Mappings

Map entire assembly to an endpoint

Or one specific type in an assembly

Or all the types in one namespace within an assembly

Demo



Publish event when order is processed

Subscribe to the event in the web application

```
public class EndpointConfig: IConfigureThisEndpoint
{
    public void Customize(
            EndpointConfiguration configuration)
    {
        configuration.UsePersistence<InMemoryPersistence>();
    }
}
```

Configuration

Relies on defaults

Combination code/config file

IConfigureThisEndpoint

INeedInitialization

```
endpointConfiguration.SendOnly()
```

Send Only Endpoint

No overhead for receiving messages

Serialization

Default: XML, soon JSON

config.UseSerialization<JsonSerializer>();

BSON, Binary or write your own

Logging

Use built-in or your favorite logging framework

Debug, Info, Warn, Error, Fatal

Written to console, trace and rolling file

Set threshold in config file

Persistence

- Used internally by NServiceBus
- No default
- InMemoryPersistence: Built into core, for testing
- NHibernatePersistence: SQL server and Oracle
- RavenDbPersistence
- AzurePersistence

```
config.UsePersistence<InMemoryPersistence,
    StorageType.Timeouts>();

config.UsePersistence<NHibernatePersistence,
    StorageType.Sagas>();

config.UsePersistence<NHibernatePersistence,
    StorageType.Outbox>();
```

Using Multiple Persistence Types

MSMQ

Native to Windows

Decentralized: each machine has its own queues

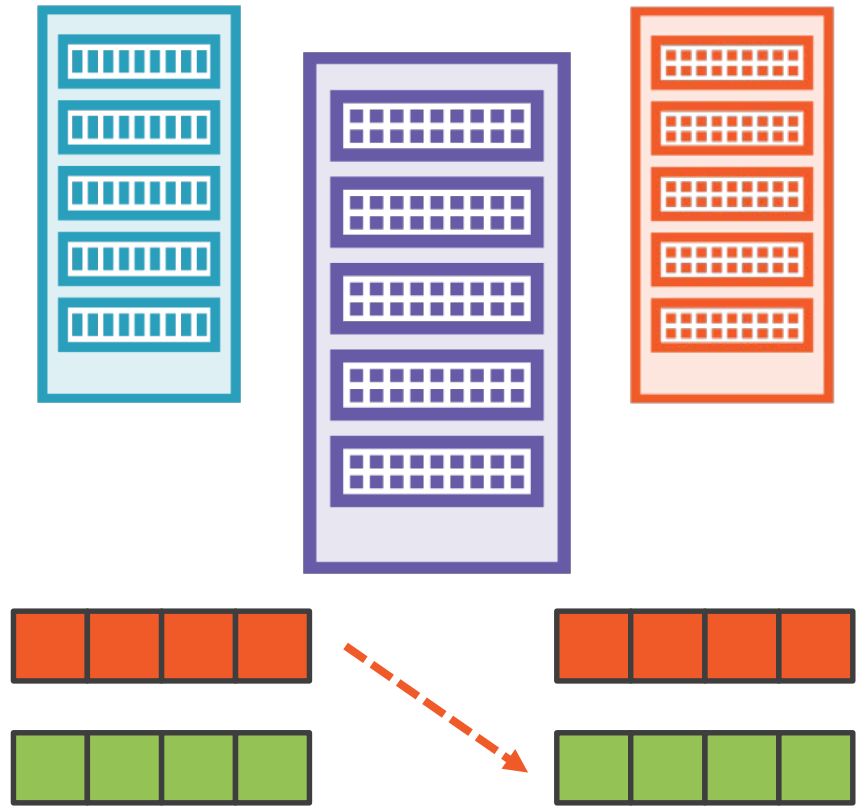
Store and forward

RabbitMQ

Broker style

Multiple platforms

Supports AMQP



SQL Server

Use table as a queue

Polling

For small projects and conversions

Windows Azure

Queues

Service Bus

Transports

Abstraction

Easy to switch

Configuration detail

```
public class Installer : INeedToInstallSomething
{

    public Task Install(string identity)
    {
    }

}
```

Installers

Depending on configuration create e.g. queues and schemas

INeedToInstallSomething interface

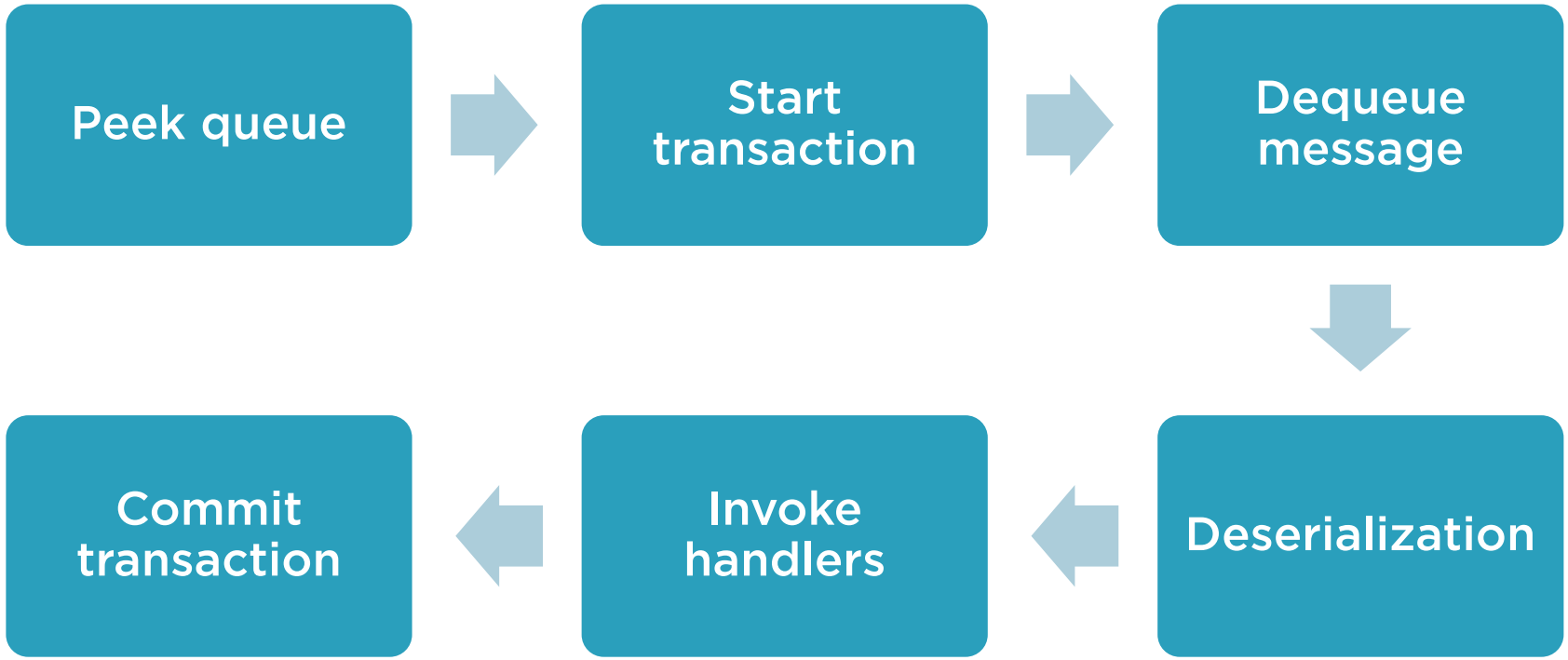
Different behaviors for debugging, self-hosting, and NServiceBus hosted

Fault Tolerance

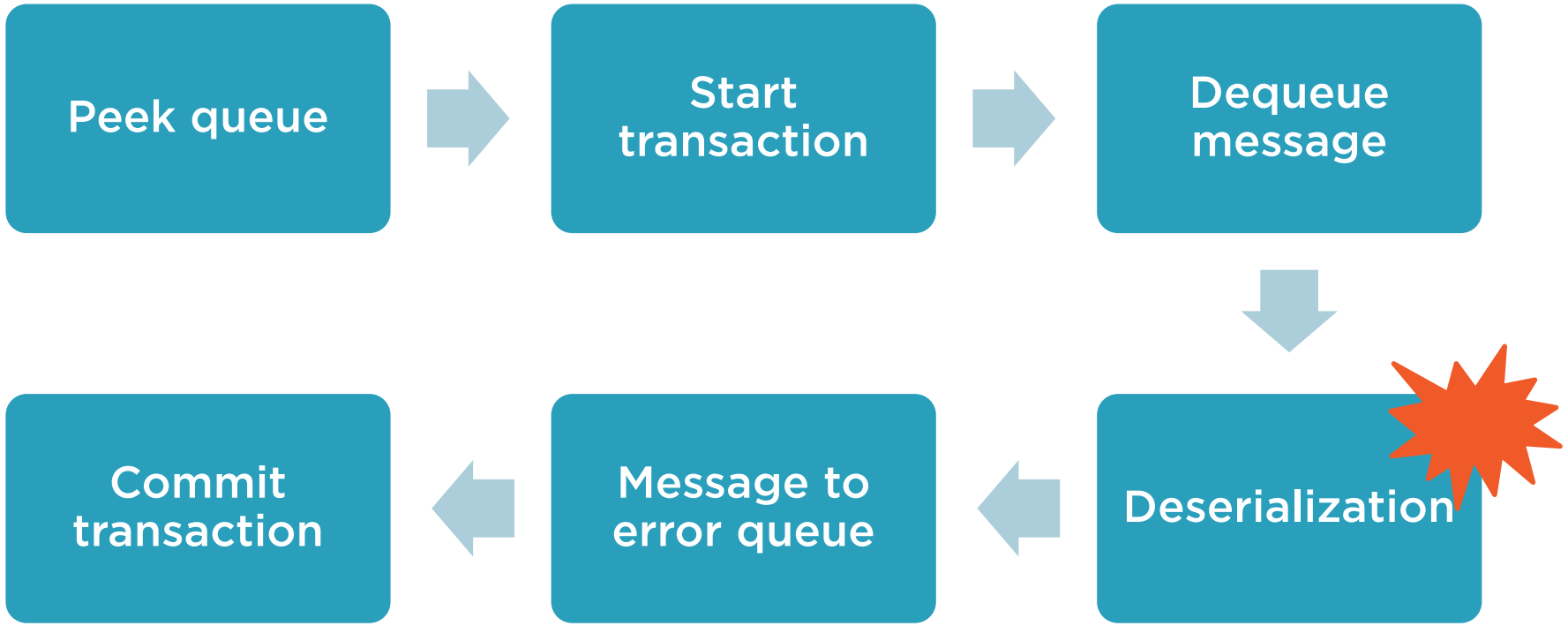
You know:
Software will fail
Infrastructure will fail

You want:
No loss of data
Resiliency

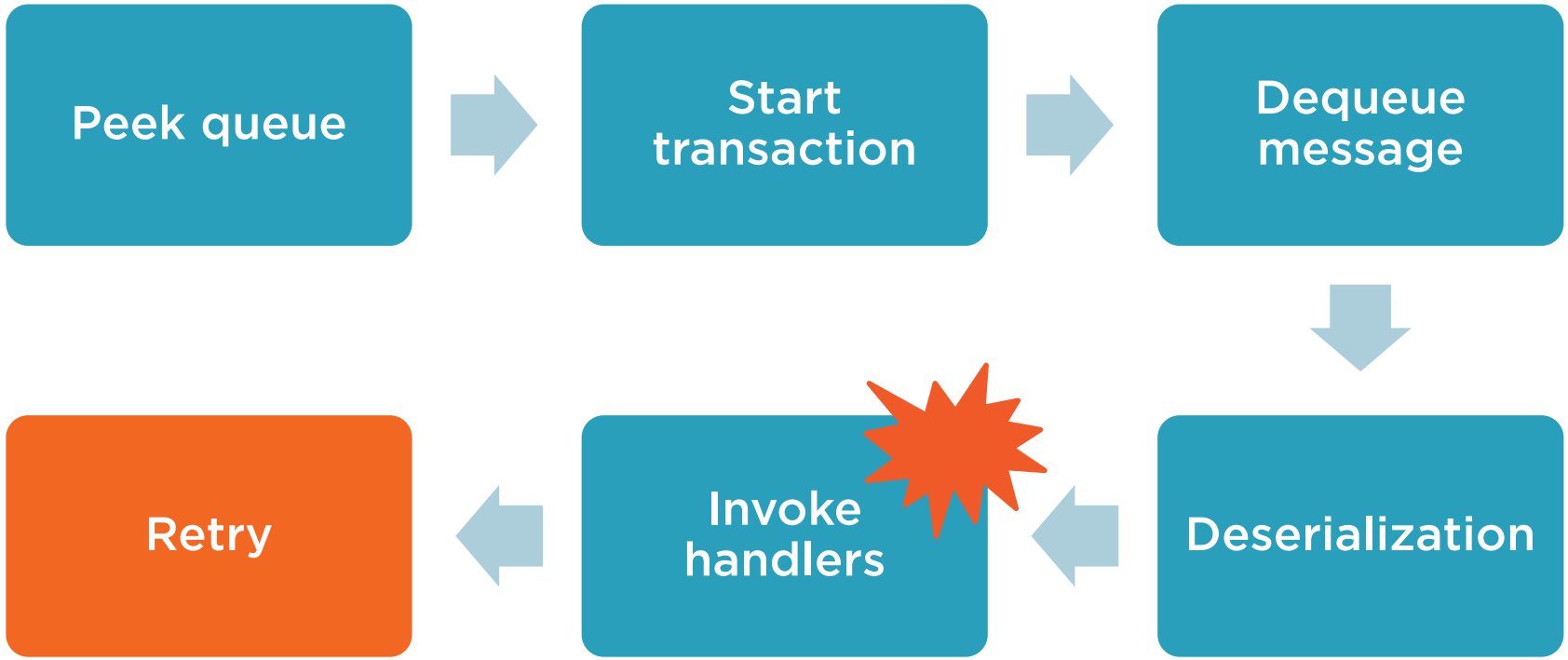
Happy Path


Peek queue
Start transaction
Dequeue message
Deserialization
Invoke handlers
Commit transaction

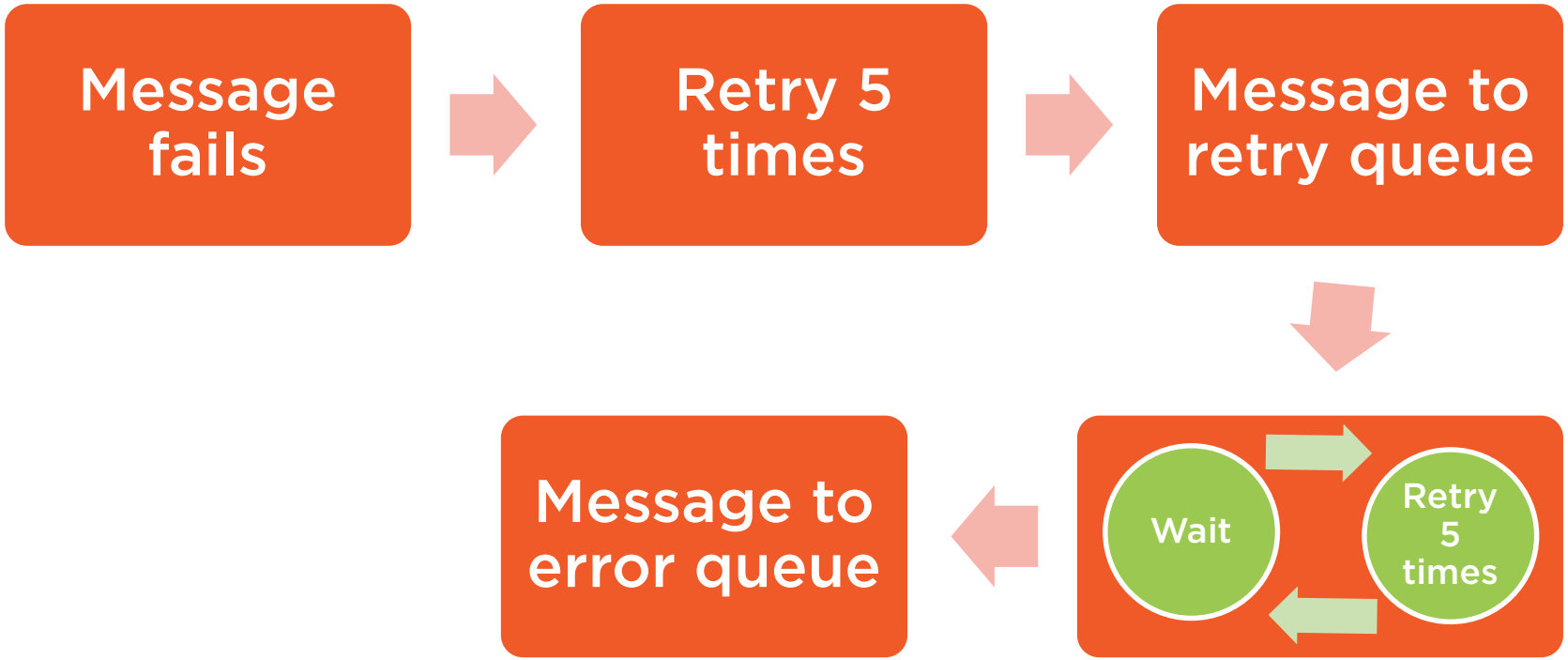
Failure During Deserialization



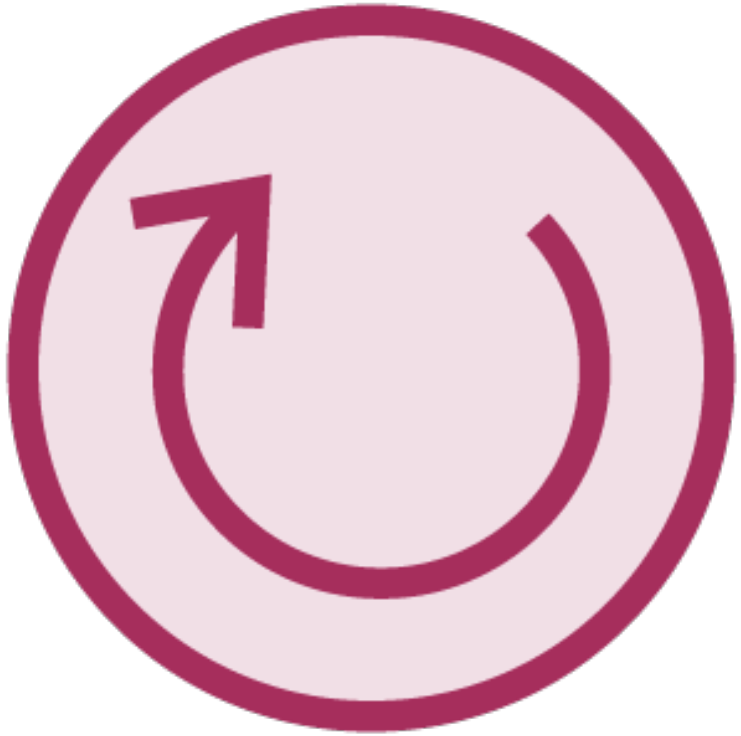
Failure of Handler(s)



Retries


Message fails
Retry 5 times
Message to retry queue
Wait
Retry 5 times
Message to error queue

Delayed Retries



Configurable: time increase and number of retries

Transient errors won't show up in error queue

Will take some time before the message is in the error queue

Error Queue

Holds messages that can’t be processed

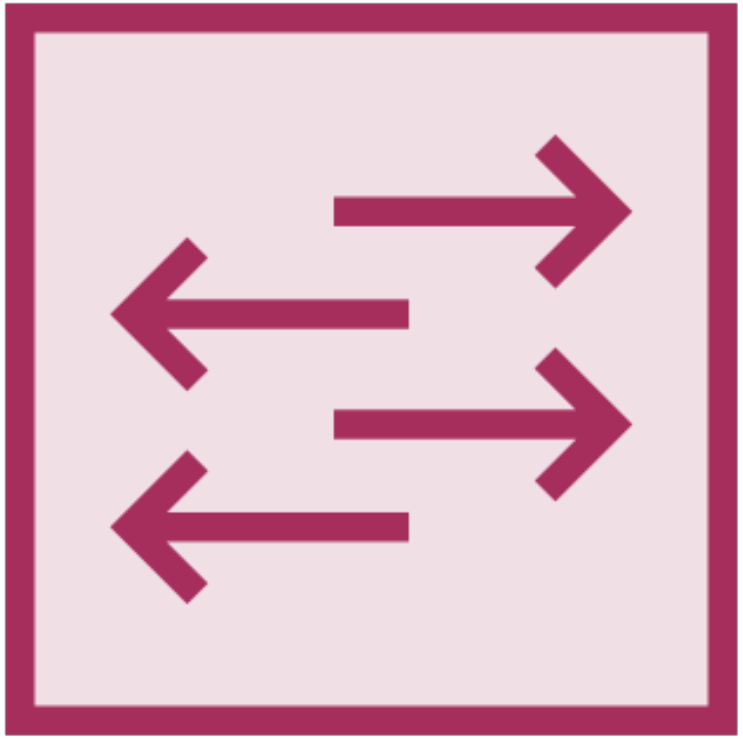
Keeps these messages out of the way

Optionally fix

Optionally replay

ServiceInsight or ServicePulse

Request/Response

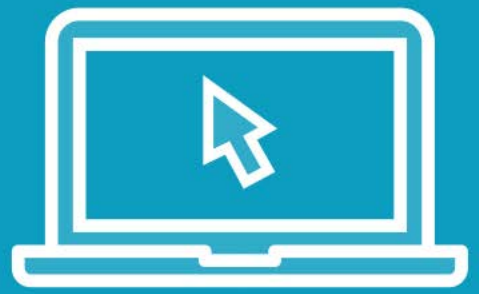


Send message and wait for response using queues

Temporal coupling

Look at alternatives

Demo



Get pricing info on the fly

Summary

NServiceBus enables messaging between applications and is highly configurable and extensible

Types of messages: commands, events, request/response

Fault tolerant