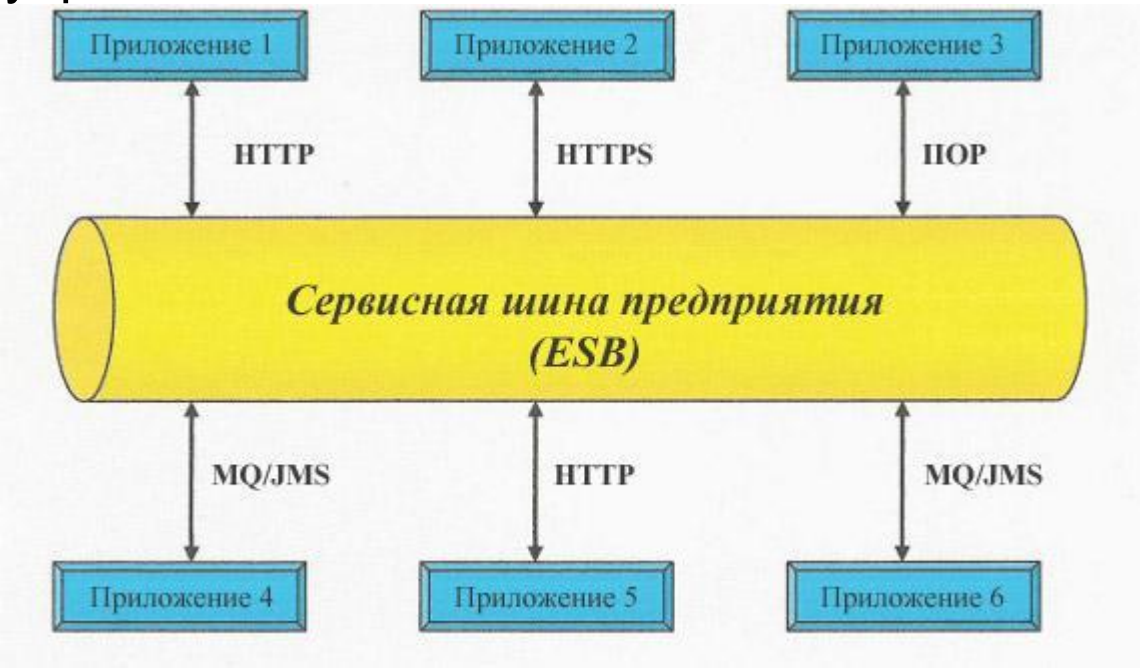
ESB RabbitMQ via MassTransit

Содержание

- Сервисная шина предприятия
- Брокер сообщений
- RabbitMq
- MassTransit
- Примеры

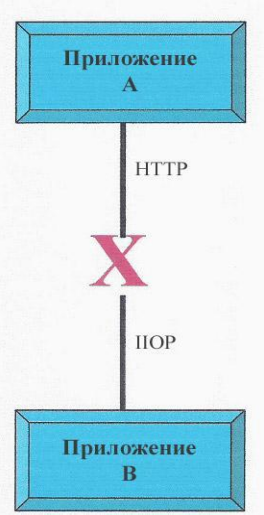
Enterprise Service Bus

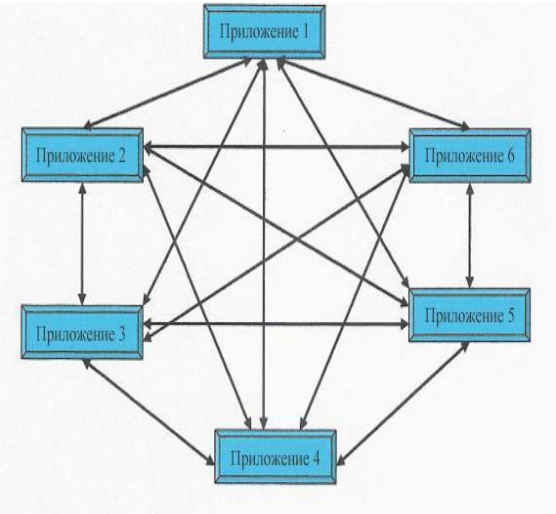
Сервисная шина предприятия – это, в первую очередь, концепция, которая позволяет интегрировать разрозненные системы в единый комплекс с централизованным управлением.

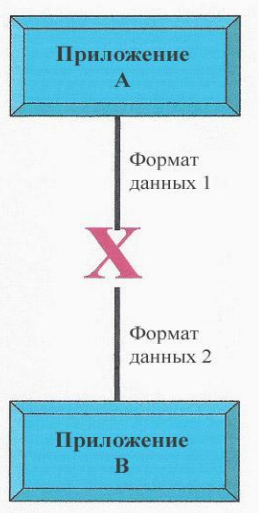


Основные возможности ESB

- Масштабируемость соединений и маршрутизации
- Преобразование протоколов
- Преобразование форматов сообщений/данных







Брокер сообщений

Проблему Масштабируемости соединений и маршрутизации можно решить брокером сообщений. Он обеспечивает гарантированную доставку сообщения от поставщика к подписчику

Брокеры:

- RabbitMQ
- Kafka
- Amazon SQS
- Apache ActiveMQ
- WebSphere MQ
- Java Message Service

Основные протоколы:

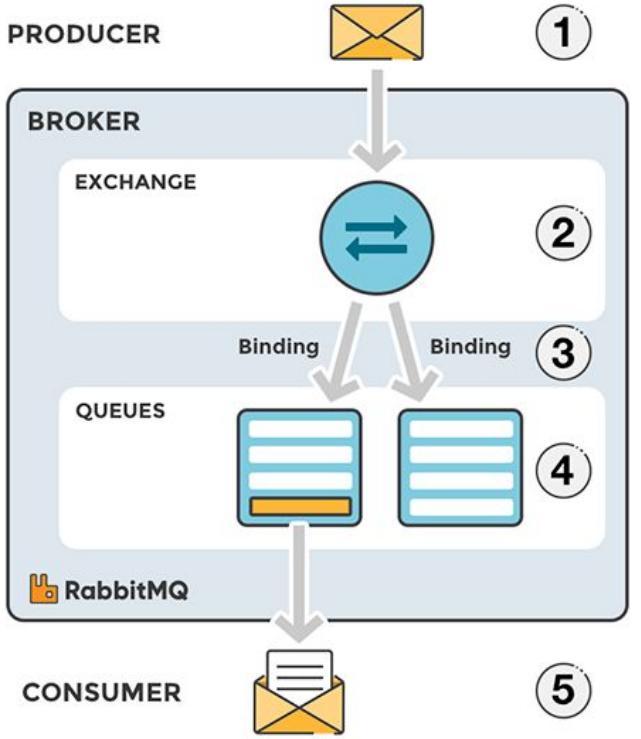
- Advanced Message Queuing Protocol (AMQP)
- STOMP
- MQ Telemetry Transport

RabbitMQ

RabbitMQ – брокер сообщений, основанный на протоколе AMQP (Advanced Message Queuing Protocol)

Базовые понятия

- Exchange – получает сообщение от поставщика и отправляет его в очередь.
- Queue - буфер который хранит сообщения
- Binding – связь между очередью и обработчиком сообщений
- Routing key – ключ на который смотрит обработчик и решает куда отправить



MassTransit

MassTransit - это библиотека с открытым исходным кодом, разработанная на языке C# для .NET платформы, упрощающая работу с шиной данных.

Возможности:

- Управление подключением
- Обработка ошибок и пропущенных сообщений
- Сериализация (JSON, BSON и XML) + шифрование (AES-256)
- Управление жизненным циклом потребителей
- Сага
- Планирование
- Мониторинг
- Модульное тестирование

Поддерживаемые шины данных:

- RabbitMq
- Azure Service Bus
- ActiveMQ
- Amazon SQS

Инициализация MassTransit

```
builder.AddMassTransit(internalContainer =>
{
    internalContainer.AddBus(container => Bus.Factory.CreateUsingRabbitMq(cfg =>
    {
        var options = container.Resolve<IBusConfiguration>();

        internalContainer.AddConsumersFromContainer(container);
        internalContainer.AddSagasFromContainer(container);

        cfg.Host(options.Url, h =>
        {
            h.Username(options.UserName);
            h.Password(options.Password);
        });

        cfg.ReceiveEndpoint(options.QueueName, ec =>
        {
            ec.ConfigureConsumers(container);
            ec.ConfigureSagas(container);

            ec.LoadStateMachineSagasCustom(container);
        });
    }));
});
```

Публикация сообщения

```
var message = new
{
    Name = Guid.NewGuid(),
    Id = Guid.NewGuid(),
    CorrelationId = Guid.NewGuid()
};

await _publishEndpoint.Publish<DistributionCreated>(message, stoppingToken);
```

Потребитель сообщения

```
3 references
public class DistributionCreatedConsumer : IConsumer<DistributionCreated>
{
    private readonly ILogger<DistributionCreatedConsumer> _logger;

    0 references
    public DistributionCreatedConsumer(ILogger<DistributionCreatedConsumer> logger)
    {
        _logger = logger;
    }

    3 references
    public async Task Consume(ConsumeContext<DistributionCreated> context)
    {
        _logger.LogInformation("Consumed::Created:: Id: {0}, Name: {1}", context.Message.Id, context.Message.Name);
    }
}
```

Saga consumer

```
0 references
public class DistributionSaga :
    ISaga,
    InitiatedBy<DistributionCreated>,
    Orchestrates<DistributionUpdated>
{
    2 references
    public Guid CorrelationId { get; set; }

    1 reference
    public Guid Id { get; set; }
    1 reference
    public string Name { get; set; }

    3 references
    public Task Consume(ConsumeContext<DistributionCreated> context)
    {
        Id = context.Message.Id;

        return Task.CompletedTask;
    }

    3 references
    public Task Consume(ConsumeContext<DistributionUpdated> context)
    {
        Name = context.Message.Name;

        return Task.CompletedTask;
    }
}
```

Saga state machine

```
1 reference
public class DistributionStateMachine :
    MassTransitStateMachine<DistributionState>
{
    2 references
    public State Active { get; private set; }
    1 reference
    public State Done { get; private set; }

    1 reference
    public Event<DistributionCreated> Started { get; private set; }
    2 references
    public Event<DistributionUpdated> Updated { get; private set; }

    0 references
    public DistributionStateMachine()
    {
        Event(() => Updated, x => x.CorrelateBy<Guid>(p => p.CorrelationId, m => m.Message.CorrelationId));

        Initially(
            When(Started)
                .Activity(x => x.OfInstanceType<PublishDistributionLoggerActivity>())
                .Then(context => context.Instance.Id = context.Data.Id)
                .ThenAsync(ActivateInstanceAsync)
                .Catch<NotImplementedException>(context => context)
                .TransitionTo(Active));

        During(Active,
            When(Updated)
                .Activity(x => x.OfInstanceType<PublishDistributionLoggerActivity>())
                .Then(UpdateInstance)
                .TransitionTo(Done)
                .Finalize());

        SetCompletedWhenFinalized();
    }
```

Activity

```
6 references
public class PublishDistributionLoggerActivity : Activity<DistributionState>
{
    readonly ConsumeContext _context;
    private readonly ILogger<PublishDistributionLoggerActivity> _logger;

    0 references
    public PublishDistributionLoggerActivity(ConsumeContext context, ILogger<PublishDistributionLoggerActivity> logger)
    {
        _context = context;
        _logger = logger;
    }

    1 reference
    public void Probe(ProbeContext context)
    {
        context.CreateScope("publisher");
    }

    0 references
    public void Accept(StateMachineVisitor visitor)
    {
        visitor.Visit(this);
    }

    0 references
    public async Task Execute(BehaviorContext<DistributionState> context, Behavior<DistributionState> next)
    {
        _logger.LogInformation($"StateMachine::Event {context.Event.Name}");

        await next.Execute(context).ConfigureAwait(false);
    }
```

Источники

- https://www.ibm.com/developerworks/ru/library/ws-universalports-esb/index.html https://habr.com/ru/post/149694/
- https://thewebland.net/development/devops/what-is-rabbitmq/
- https://www.rabbitmq.com/
- https://habr.com/ru/post/64192/
- https://masstransit-project.com