perplexity

Посмотри эти два кода.

Что-то subscriber аварийно завершается
Код раньше работал
using MassTransit;
using MassTransitFastMessages;

// Конфигурация для максимальной скорости
var busControl = Bus.Factory.CreateUsingRabbitMq(cfg ⇒
{
cfg.Host("rabbitmq://localhost", h ⇒
{
h.Username("guest");
h.Password("guest");
});
});

await busControl.StartAsync();

try
{
// Получаем endpoint для прямой отправки (быстрее чем Publish)
var endpoint = await busControl.GetSendEndpoint(new Uri("rabbitmq://localhost/fast-candlestick-queue"));

```
Console.WriteLine("Fast Publisher started. Choose generator:");
Console.WriteLine("1 - Random Generator");
Console.WriteLine("2 - High Performance Generator");

var choice = Console.ReadLine();

ICandleStickGenerator generator = choice == "2"
    ? new HighPerformanceGenerator()
    : new RandomCandleStickGenerator();

long messageCount = 0;
var timer = System.Diagnostics.Stopwatch.StartNew();

generator.OnNewCandleStick += async (candle) =>
{
    try
    {
        // Прямая отправка в endpoint - максимальная скорость
        await endpoint.Send(candle, context =>
```

```
        {
            context.Durable = false; // Неустойчивые сообщения
        }).ConfigureAwait(false);

        messageCount++;
        if (messageCount % 100 == 0)
        {
            var elapsed = timer.Elapsed.TotalSeconds;
            var rate = messageCount / elapsed;
            Console.WriteLine($"Sent {messageCount} messages | Rate: {rate:F2} msg/sec");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Send error: {ex.Message}");
    }
};

generator.Start();

Console.WriteLine("Generator started. Press any key to stop...");
Console.ReadKey();

generator.Stop();

timer.Stop();
Console.WriteLine($"Total sent: {messageCount} messages in {timer.Elapsed.TotalSeconds:F2
Console.WriteLine($"Average rate: {messageCount / timer.Elapsed.TotalSeconds:F2} msg/sec'
```

}
finally
{
await busControl.StopAsync();
}

using MassTransit;
using MassTransitFastMessages;
using MassTransitFastSubscriber;
using System.Threading.Channels;

// Создаем высокопроизводительный channel для обработки сообщений
var channel = Channel.CreateUnbounded<CandleStick>(new UnboundedChannelOptions
{
SingleReader = false,
SingleWriter = true
});

// Запускаем фоновую задачу для обработки сообщений из channel
_ = Task.Run(async () ⇒
{
await foreach (var candle in channel.Reader.ReadAllAsync())

{
// Быстрая обработка
Console.WriteLine($"Received: {candle.Ticker} | O:{candle.Open} H:{candle.High} L:
{candle.Low} C:{candle.Close}");
}
});

// Конфигурация MassTransit для максимальной скорости приема
var busControl = Bus.Factory.CreateUsingRabbitMq(cfg ⇒
{
cfg.Host("rabbitmq://localhost", h ⇒
{
h.Username("guest");
h.Password("guest");
});

```
// Настройка endpoint для максимальной производительности
cfg.ReceiveEndpoint("fast-candlestick-queue", e =>
{
    // Неустойчивая очередь для скорости
    e.Durable = false;
    e.AutoDelete = true;

    // Настройки производительности
    e.PrefetchCount = 100;
    e.ConcurrentMessageLimit = 10;

    // Отключаем повторные отправки для скорости
    e.UseMessageRetry(r => r.None());

    e.Consumer(() => new FastCandleStickConsumer(channel.Writer));
});
```

});

await busControl.StartAsync();

try
{
Console.WriteLine("Fast Subscriber started. Processing messages...");
Console.WriteLine("Press any key to exit...");
Console.ReadKey();
}
finally
{
channel.Writer.Complete();
await busControl.StopAsync();
}

Может быть очереди неправильно написаны
?
Начинает работать производитель. Я его вижу в Rabbit, даже цифры активности меняются.
Subscriber запускается и ничего не показывает.
Через некоторое время вываливается в exception
Контейнер вроде работает

Да, проблема, скорее всего, в неправильной конфигурации очереди и обмена (exchange) между отправителем и подписчиком. У MassTransit при `Send` и `ReceiveEndpoint` имена должны точно совпадать **в терминах endpoint и exchange**, а у тебя сейчас есть несколько критических несоответствий, из-за которых сообщения просто не попадают к consumer.

Почему не работает

1. **MassTransit всегда публикует в exchange**, а не напрямую в очередь.
   В случае вызова

   ```
   await endpoint.Send(candle);
   ```

   MassTransit создает exchange с именем `"fast-candlestick-queue"` и отправляет туда сообщения, но consumer (`ReceiveEndpoint("fast-candlestick-queue", ...)`) создаёт отдельный собственный exchange и очередь с тем же именем, **но не связывает их между собой**. RabbitMQ по умолчанию не делает эту привязку без инструкции MassTransit.[1] [2]

2. `ConfigureConsumeTopology` **по умолчанию выключен** в ручных конфигурациях.
   Поэтому exchange для `CandleStick` не привязывается к целевой очереди — сообщения есть в брокере, но идут "в никуда".[2]

3. `Durable = false` **и** `AutoDelete = true` делают очередь временной.
   Она исчезает при перезапуске subscriber. Если producer стартует раньше, сообщений просто некуда доставлять — очередь ещё не создана.[2]

4. **Consumer, вероятно, не вызывается вовсе**, и через несколько секунд MassTransit выбрасывает `TaskCanceledException` или `ObjectDisposedException`, если `busControl` останавливается при пустом канале.[3]

Как исправить

Вариант 1 — использовать `Publish` и `Subscribe` (рекомендовано MassTransit)

В продюсере:

```
await busControl.Publish(candle);
```

В подписчике:

```
cfg.ReceiveEndpoint("fast-candlestick-queue", e =>
{
    e.ConfigureConsumeTopology = true;  // важно!
    e.Consumer(() => new FastCandleStickConsumer(channel.Writer));
});
```

Так exchange для `CandleStick` будет создан автоматически и связан с твоей очередью.

Вариант 2 — остаться с `Send`, но точно указать endpoint имя

Ты сейчас делаешь:

```
await busControl.GetSendEndpoint(new Uri("rabbitmq://localhost/fast-candlestick-queue"));
```

MassTransit ожидает "Logical Endpoint", зарегистрированный через `ReceiveEndpoint`, а не просто имя очереди.
Для этого нужно добавить уникальный `QueueName` **и consumer имеет работать как именно endpoint**:

```
cfg.ReceiveEndpoint("fast-candlestick-queue", e =>
{
    e.BindMessageExchanges = false;
    e.ConfigureConsumeTopology = false;
    e.Consumer(() => new FastCandleStickConsumer(channel.Writer));
});
```

И producer должен обращаться к точно такому URI, но с правильной схемой:

```
await busControl.GetSendEndpoint(new Uri("rabbitmq://localhost/fast-candlestick-queue?du
```

Рекомендуемая проверка

1. Зайди в RabbitMQ Management → "Exchanges".
   Убедись, что exchange `fast-candlestick-queue` привязан к одноименной `queue`.
2. Запусти сначала subscriber, **потом publisher**.
   Так очередь успеет создаться.
3. Удали `AutoDelete=true` на время отладки, иначе очередь исчезнет при остановке subscriber.

Если всё сделать так, consumer начнёт получать `CandleStick`, а частота логов «Received: ...» стабильно будет соответствовать скорости публикации.

⁂

У меня в Exchange есть строка

MassTransitFastMessages:CandleStick
и ниже еще одна строка
fast-candlestick-queue

Это поведение полностью соответствует типичной топологии MassTransit в RabbitMQ и объясняет, почему твой subscriber не получает сообщения.

Что обозначают эти две строки в RabbitMQ

1. `MassTransitFastMessages:CandleStick` — это **exchange**, автоматически созданный MassTransit для твоего типа сообщения `CandleStick`.
   Exchange называется по полному имени типа (`Namespace:ClassName`). Это часть *publish topology* MassTransit.[11] [12] [13]
   Каждый вызов `Publish(CandleStick)` отправляет сообщение в этот exchange.
   Если же ты используешь `Send`, то exchange под конкретный тип не используется, и сообщение направляется в endpoint напрямую.
2. `fast-candlestick-queue` — это **очередь и связанная с ней exchange** (MassTransit создаёт для каждой `ReceiveEndpoint` собственный exchange с тем же именем).
   Эта очередь принимает сообщения **только если есть привязка (binding)** между exchange `MassTransitFastMessages:CandleStick` и `fast-candlestick-queue`.
   Без этой связи сообщения оседают в exchange `MassTransitFastMessages:CandleStick` и не доставляются подписчику.[13] [11]

Почему твой subscriber ничего не принимает

- Producer отправляет сообщения **в exchange** `MassTransitFastMessages:CandleStick`, но consumer (`fast-candlestick-queue`) слушает свою очередь, **не связанную с этим exchange**.

- В RabbitMQ Management ты видишь оба объекта, но между ними **нет binding**, поэтому сообщений в очередь не попадает.
- Через некоторое время приложение вылетает, потому что consumer простаивает, а MassTransit останавливает pipeline.

Как исправить

Вариант 1 — использовать `Publish` вместо `Send` (рекомендуется)

Когда producer вызывает:

```
await busControl.Publish(candle);
```

MassTransit автоматически создаёт и настраивает нужный binding между `MassTransitFastMessages:CandleStick` и `fast-candlestick-queue`.
Consumer при этом можно оставить без изменений:

```
cfg.ReceiveEndpoint("fast-candlestick-queue", e =>
{
    e.ConfigureConsumeTopology = true; // важно!
    e.Consumer(() => new FastCandleStickConsumer(channel.Writer));
});
```

Вариант 2 — вручную создать binding, если хочешь оставить `Send`

Если ты сознательно хочешь использовать прямую отправку через `Send`, привяжи очередь к exchange вручную в RabbitMQ или добавь конфигурацию:

```
cfg.ReceiveEndpoint("fast-candlestick-queue", e =>
{
    e.Bind("MassTransitFastMessages:CandleStick"); // вручную связывает exchange типа coc
    e.Consumer(() => new FastCandleStickConsumer(channel.Writer));
});
```

Итог:
То, что ты видишь в RabbitMQ Management (exchange с типом `MassTransitFastMessages:CandleStick` и очередь `fast-candlestick-queue`), — это не ошибка, а стандартный результат разных топологий.
Главное сейчас — **создать binding** между ними или перейти на `Publish`, чтобы MassTransit сделал это автоматически.

⁂

await busControl.Publish(candle); - это не помогло.

Вот что пишет docker
2025-10-22 07:27:43.741928+00:00 [error] <0.2006.0> closing AMQP connection 172.17.0.1:58644 → 172.17.0.2:5672 (duration: '0ms'):
2025-10-22 07:27:43.741928+00:00 [error] <0.2006.0> HTTP GET request detected on AMQP port. Ensure the client is connecting to the correct port.
2025-10-22 07:27:44.803930+00:00 [info] <0.2012.0> accepting AMQP connection 172.17.0.1:58652 → 172.17.0.2:5672
2025-10-22 07:27:44.804154+00:00 [error] <0.2012.0> closing AMQP connection 172.17.0.1:58652 → 172.17.0.2:5672 (duration: '1s'):
2025-10-22 07:27:44.804154+00:00 [error] <0.2012.0> HTTP GET request detected on AMQP port. Ensure the client is connecting to the correct port.
2025-10-22 07:27:49.828089+00:00 [info] <0.2016.0> accepting AMQP connection 172.17.0.1:58658 → 172.17.0.2:5672
2025-10-22 07:27:49.828316+00:00 [error] <0.2016.0> closing AMQP connection 172.17.0.1:58658 → 172.17.0.2:5672 (duration: '6s'):
2025-10-22 07:27:49.828316+00:00 [error] <0.2016.0> HTTP GET request detected on AMQP port. Ensure the client is connecting to the correct port.
2025-10-22 07:27:54.809935+00:00 [info] <0.2022.0> accepting AMQP connection 172.17.0.1:58672 → 172.17.0.2:5672
2025-10-22 07:27:54.810091+00:00 [error] <0.2022.0> closing AMQP connection <0.2022.0> (172.17.0.1:58672 → 172.17.0.2:5672, duration: '10s'):
2025-10-22 07:27:54.810091+00:00 [error] <0.2022.0> {handshake_timeout,handshake}
2025-10-22 07:27:59.845922+00:00 [info] <0.2029.0> accepting AMQP connection 172.17.0.1:33012 → 172.17.0.2:5672
2025-10-22 07:27:59.846064+00:00 [error] <0.2029.0> closing AMQP connection <0.2029.0> (172.17.0.1:33012 → 172.17.0.2:5672, duration: '10s'):
2025-10-22 07:27:59.846064+00:00 [error] <0.2029.0> {handshake_timeout,handshake}
2025-10-22 07:28:00.316013+00:00 [error] <0.1835.0> closing AMQP connection <0.1835.0> (172.17.0.1:45918 → 172.17.0.2:5672 - MassTransitFastPublisher, duration: '4M, 0s'):
2025-10-22 07:28:00.316013+00:00 [error] <0.1835.0> missed heartbeats from client, timeout: 60s

Из логов RabbitMQ видно, что проблема не в MassTransit, а в самой сети или некорректном подключении клиента к **не тому порту** RabbitMQ. Ошибки «HTTP GET request detected on AMQP port» и «handshake_timeout» однозначно указывают на ошибочные попытки подключения к **порту AMQP (5672)** как к **HTTP-порту**.[21] [22] [23]

Что реально происходит

1. **HTTP GET request detected on AMQP port**
   Эта ошибка возникает, если клиент пытается подключиться по HTTP (или REST-запросом) к порту 5672, который предназначен только для протокола AMQP.
   Обычно это происходит, когда:
   - подключение идёт к `http://localhost:5672` вместо `rabbitmq://localhost` в настройках MassTransit;
   - в Docker контейнере перепутаны порты (`15672` — это Management UI, `5672` — AMQP);
   - MassTransit настроен с `cfg.Host("rabbitmq://localhost")`, но docker-контейнер использует другой hostname или порт.
2. **{handshake_timeout, handshake}**
   RabbitMQ принял TCP-соединение, но клиент не завершил AMQP handshake — чаще всего это значит, что клиент завис до установления сессии из-за сетевой конфигурации (например, неверный адрес или порт в `cfg.Host`).[22] [24]
3. **missed heartbeats from client**
   После установления соединения RabbitMQ не получает heartbeat-сообщения от клиента в течение 60 секунд.
   Это обычно вызвано:
   - зависанием клиента;
   - проблемами сети внутри Docker bridge;
   - неправильно настроенным параметром heartbeat или задержками в обработке.[23] [25] [26]

Как исправить

1. Проверь порты и URI

Открой `docker ps` и посмотри порты RabbitMQ — по умолчанию должно быть:

```
5672:5672    # AMQP
15672:15672  # Web UI
```

В коде **MassTransit** клиент должен подключаться вот так:

```
cfg.Host("localhost", "/", h =>
{
    h.Username("guest");
    h.Password("guest");
});
```

Важно: не используйте `http://`, только строку `"localhost"` и схему `rabbitmq://` при создании URI вручную (например, в `GetSendEndpoint`).

2. Если используешь docker-compose

Поменяй `Host` на имя контейнера RabbitMQ, например:

```
cfg.Host("rabbitmq", "/", h =>
{
    h.Username("guest");
    h.Password("guest");
});
```

и убедись, что сервис `rabbitmq` объявлен в `docker-compose.yml`.

3. Проверь firewall и delays

Запусти:

```
docker exec -it rabbitmq ping <имя_контейнера_приложения>
```

Если пинг не проходит — проблема с сетью.
Network mode должен быть `bridge` или общая сеть `docker network`.

4. Убедись, что Publisher и Subscriber используют один и тот же Virtual Host `/`

Иначе producer может публиковать в `/`, а subscriber слушать `test` или другой виртуальный хост.

Другими словами, сейчас твой RabbitMQ получает HTTP-запросы на порт AMQP. Нужно убедиться, что код MassTransit соединяется именно через протокол AMQP (`5672`) без `http://`, по имени контейнера RabbitMQ, и что Publisher и Subscriber находятся в одной docker-сети с одинаковым virtual host.

⁂

MassTransit.RabbitMqConnectionException

HResult=0x80131500
Message=ReceiveTransport faulted: guest@localhost:5672/
Source=MassTransit
StackTrace:
at MassTransit.Transports.ReceiveTransport`1.ReceiveTransportAgent.<RunTransport>d__10.MoveNext() in /_/src/MassTransit/Transports/ReceiveTransport.cs:line 216 at MassTransit.Transports.ReceiveTransport`1.ReceiveTransportAgent.<Run>d__9.MoveNext() in *//src/MassTransit/Transports/ReceiveTransport.cs:line 145*
at MassTransit.Transports.StartHostHandle.<EndpointsReady>d__8.MoveNext() in

//src/MassTransit/Transports/StartHostHandle.cs:line 45
at MassTransit.Transports.StartHostHandle.<ReadyOrNot>d__7.MoveNext() in
//src/MassTransit/Transports/StartHostHandle.cs:line 35
at MassTransit.MassTransitBus.Handle.<ReadyOrNot>d__11.MoveNext() in
//src/MassTransit/MassTransitBus.cs:line 415
at MassTransit.MassTransitBus.<StartAsync>d__30.MoveNext() in
//src/MassTransit/MassTransitBus.cs:line 209
at MassTransit.MassTransitBus.<StartAsync>d__30.MoveNext() in
//src/MassTransit/MassTransitBus.cs:line 247
at Program.<<Main>$>d__0.MoveNext() in
F:\Projects\VS\MassTransitFastMessagesSln\MassTransitFastSubscriber\Program.cs:line 50
at Program.<Main>(String[] args)

This exception was originally thrown at this call stack:
System.ThrowHelper.ThrowObjectDisposedException(object)
System.Threading.SemaphoreSlim.Release(int)
RabbitMQ.Client.Impl.Channel.QueueDeclareAsync(string, bool, bool, bool,
System.Collections.Generic.IDictionary<string, object>, bool, bool,
System.Threading.CancellationToken)
MassTransit.RabbitMqTransport.ScopeChannelContext.QueueDeclare(string, bool, bool, bool,
System.Collections.Generic.IDictionary<string, object>, System.Threading.CancellationToken) in
ScopeChannelContext.cs
MassTransit.RabbitMqTransport.Middleware.ConfigureRabbitMqTopologyFilter<TSettings>.Decl
are(MassTransit.RabbitMqTransport.ChannelContext,
MassTransit.RabbitMqTransport.Topology.Queue, System.Threading.CancellationToken) in
ConfigureRabbitMqTopologyFilter.cs
MassTransit.RabbitMqTransport.Middleware.ConfigureRabbitMqTopologyFilter<TSettings>.Confi
gureTopology(MassTransit.RabbitMqTransport.ChannelContext,
System.Threading.CancellationToken) in ConfigureRabbitMqTopologyFilter.cs
MassTransit.OneTimeSetupMethod.SetupPayload.__SetupAsync|0() in OneTimeSetupMethod.cs
MassTransit.PipeExtensions.OneTimeSetup<T>(MassTransit.PipeContext,
MassTransit.OneTimeSetupCallback) in PipeExtensions.cs
MassTransit.OneTimeContextPayload<TPayload>.RunOnce(MassTransit.OneTimeSetupMethod)
in OneTimeContextPayload.cs
MassTransit.RabbitMqTransport.Middleware.ConfigureRabbitMqTopologyFilter<TSettings>.Confi
gure(MassTransit.RabbitMqTransport.ChannelContext, System.Threading.CancellationToken) in
ConfigureRabbitMqTopologyFilter.cs
...
[Call Stack Truncated]

Inner Exception 1:
ObjectDisposedException: Cannot access a disposed object.
Object name: 'System.Threading.SemaphoreSlim'.
Вот exception
Может быть Вы разберетесь

Ошибка `MassTransit.RabbitMqConnectionException` с внутренним `ObjectDisposedException` и сообщением `ReceiveTransport faulted: guest@localhost:5672/` — это типичная ситуация, когда MassTransit не может создать или синхронизировать объекты RabbitMQ из-за **конфликтующих параметров существующих обменников/очередей** и **разорванного канала**.
Подробный анализ показывает, что это ошибка на уровне RabbitMQ-топологии, а не логики C#. [31] [32] [33]

Что происходит

1. `ObjectDisposedException: Cannot access a disposed object (System.Threading.SemaphoreSlim)`
   Это следствие того, что MassTransit закрыл соединение, когда создание очереди или обменника (`QueueDeclareAsync`) не удалось. Обычно это происходит в момент реконфигурации `ReceiveEndpoint` или при изменении свойств очереди (например, durable / autoDelete). [34] [31]
2. В стеке виден `ConfigureRabbitMqTopologyFilter.Declare` — именно этот фильтр отвечает за синхронизацию топологии (bindings, exchanges, queues).
   RabbitMQ отверг операцию из-за несоответствия существующих параметров: например, если ранее была создана **durable очередь**, а сейчас ты пытаешься создать **временную (durable=false)**.

Частая первопричина

RabbitMQ не разрешает «переопределять» уже созданные объекты с другими параметрами.
Например:

- если в прошлый раз `fast-candlestick-queue` создавалась **как durable**,
- а теперь в коде `e.Durable = false`,

RabbitMQ вернёт ошибку `PRECONDITION_FAILED` и закроет соединение. MassTransit интерпретирует это как `ReceiveTransport faulted`. [31]

Как исправить

1. **Удалить старые объекты из RabbitMQ UI**
   Войдите по адресу `http://localhost:15672`, вкладка **Queues** → удалить `fast-candlestick-queue`.
   Вкладка **Exchanges** → удалить `fast-candlestick-queue` и `MassTransitFastMessages:CandleStick`.
   Это принудительно очистит топологию, и MassTransit пересоздаст её при запуске.
2. **Следите за параметрами durable / autoDelete**
   В consumer-коде сейчас указано:

```
e.Durable = false;
e.AutoDelete = true;
```

Если в будущем хочешь стабильное поведение, можно вернуть к:

```
e.Durable = true;
e.AutoDelete = false;
```

(Это предотвратит повторные конфликты при рестартах.)

3. **Перезапусти RabbitMQ (в Docker)**

```
docker restart <rabbitmq-container-name>
```

Это удалит временные объекты, которые могли сохраниться в памяти.

4. **Не меняй пространство имён сообщений (`MassTransitFastMessages`)**
Если ранее тип `CandleStick` был объявлен как `internal` или без неймспейса, а теперь `public`, RabbitMQ распознаёт обменник как новый и сталкивает старые настройки. Всегда держи тип сообщений стабильным, иначе придётся чистить брокера.[31]

Итог

Ошибка вызвана конфликтом параметров (durable/autoDelete/public contract) между старой конфигурацией RabbitMQ и новой схемой MassTransit.
Лечение простое:

1. Очисти все exchange/queue `fast-candlestick-*` и `MassTransitFastMessages:CandleStick`.
2. Проверь, что publisher и subscriber используют одинаковый virtual host `/`, порт `5672`, и у обоих `e.Durable = true` (для начала).
3. Перезапусти RabbitMQ и приложения.

После этого `ReceiveTransport faulted` и `ObjectDisposedException` исчезнут, и consumer начнёт получать сообщения корректно.

⁂

PS C:\Users\Administrator> docker restart <rabbitmq-container-name>

At line:1 char:16

- docker restart <rabbitmq-container-name>
-
```
            ~
```

The '<' operator is reserved for future use.
+ CategoryInfo : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : RedirectionNotSupported

Ошибка PowerShell `"The '<' operator is reserved for future use"` возникает потому, что угловые скобки (`< >`) интерпретируются как **операторы перенаправления ввода**, а не как часть аргумента команды.[41] [42]

Как должно быть правильно

В PowerShell **нельзя писать** так:

```
docker restart <rabbitmq-container-name>
```

Угловые скобки — это всего лишь пример синтаксиса из документации (placeholder), а не часть команды.
Ты должен подставить реальное имя контейнера **без скобок**.

Примеры правильных команд

Если контейнер называется `rabbitmq`, нужно выполнять:

```
docker restart rabbitmq
```

Или, если хочешь узнать имя контейнера:

```
docker ps
```

(в выводе найди колонку *NAMES*, напр. `massrabbit`, `rabbitmq-fast` и т.д.)

и затем перезапусти его:

```
docker restart massrabbit
```

Альтернатива

Чтобы перезапустить **все контейнеры** сразу:

```
docker restart $(docker ps -q)
```

(На Windows $() также работает в PowerShell и выполняет командную подстановку.)

Таким образом, скобки «< >» служат в документации как обозначение переменной, а не как часть команды — их нужно убрать и заменить именем твоего контейнера.

⁂

throw await NotifyFaulted(exception, "ReceiveTransport faulted: ").ConfigureAwait(false);

Вот на этом падает
В начале удалил все что Вы сказали

Если исключение возникает на строке
`throw await NotifyFaulted(exception, "ReceiveTransport faulted: ")`,
значит, **MassTransit не может корректно запустить ReceiveEndpoint** — проблема именно в уровне подключения к RabbitMQ или в том, как конфигурируется очередь.[51] [52] [53]

Что означает `NotifyFaulted`

Эта строка находится внутри исходников `ReceiveTransportAgent` и вызывается, когда при создании транспортного канала (`ReceiveEndpoint`) происходит исключение в одном из этапов:

- подключение к RabbitMQ не удалось;
- `QueueDeclare` или `ExchangeBind` выбросили ошибку;
- канал уже закрыт (disposed) или соединение прерывистое.[54] [55]

Возможные причины

1. **Очередь или exchange уже существует с другими параметрами**
   Это по-прежнему самая частая причина:
   если `fast-candlestick-queue` осталась с прежними настройками (`durable=true`, теперь `false`), RabbitMQ возвращает `PRECONDITION_FAILED`, а MassTransit оборачивает это в `RabbitMqConnectionException`.
   Даже если в UI ты удалил очередь, RabbitMQ иногда кэширует обменники — после удаления **обязательно перезапусти контейнер**:
   ```
   docker restart rabbitmq
   ```
2. **Namespace типа `CandleStick` изменился**
   На уровне RabbitMQ exchange называется `MassTransitFastMessages:CandleStick`.
   Если ты менял namespace, сборку или visibility (`internal` → `public`), RabbitMQ считает это **новым типом**, и топология ломается.[52]

3. **Конфликт durable/autoDelete во время реконфигурации**
   Если инстанс уже создал временную очередь (`AutoDelete=true`), следующий запуск с `AutoDelete=false` или наоборот метнёт `ObjectDisposedException` через `NotifyFaulted`.[53] [52]

4. **RabbitMQ-контейнер не принял handshake**
   Иногда при запуске publisher быстрее, чем поднят `consumer`, сессия отваливается по тайм-ауту handshake.[51]
   Это решается добавлением короткой задержки перед `StartAsync()`:

```
await Task.Delay(1000);
await busControl.StartAsync();
```

Что сделать сейчас

1. **Останови оба приложения** (publisher и subscriber).
2. **Очисти RabbitMQ полностью** через UI (`Queues`, `Exchanges`, `Connections` → удалить всё, связанное с `candlestick`).
   Затем **перезапусти контейнер**:

```
docker restart rabbitmq
```

3. Проверь, что **в коде consumer**:

```
e.Durable = true;
e.AutoDelete = false;
e.ConfigureConsumeTopology = true;
```

   и **всё остальное временно закомментировано** для чистого запуска.

4. **Запусти сначала subscriber**, потом publisher.
5. Если всё стартует без исключений — можно постепенно вернуть настройки производительности (`PrefetchCount`, `AutoDelete`, и т.п.).

Иными словами, `NotifyFaulted` — это лишь последняя точка, где MassTransit сообщает о неудачном поднятии транспорта. Ошибка не в нём самом, а в том, что **топология очередей и обменников снова конфликтует** либо соединение с RabbitMQ не установилось корректно. Полная очистка брокера и одинаковые флаги `durable/autoDelete` обычно решают проблему.

⁂

1. https://stackoverflow.com/questions/56046802/masstransit-rabbitmq-why-so-many-messages-get-skipped
2. https://masstransit.io/documentation/configuration
3. https://stackoverflow.com/questions/68008955/masstransit-consumer-not-consuming-messages-when-doing-dependency-injection
4. https://github.com/MassTransit/MassTransit/discussions/5556

5. https://github.com/MassTransit/MassTransit/discussions/5301
6. https://groups.google.com/g/masstransit-discuss/c/_bnDRDGWLcc
7. https://masstransit.io/documentation/configuration/transports/rabbitmq
8. https://www.reddit.com/r/dotnet/comments/1o9txsz/masstransit_publishsubscribe_with_rabbitmq/
9. https://www.looselycoupledlabs.com/2014/07/error-handling-in-masstransit-consumers/index.htm
10. https://github.com/MassTransit/MassTransit/issues/2072?timeline_page=1
11. https://masstransit.io/documentation/configuration/topology/message
12. https://masstransit.io/documentation/transports/rabbitmq
13. https://bartwullems.blogspot.com/2018/09/masstransitchange-exchange-naming.html
14. https://habr.com/ru/articles/758938/
15. https://stackoverflow.com/questions/69078000/how-do-i-set-a-default-naming-convention-for-subscriptions-and-queues-in-masstra
16. https://github.com/MassTransit/MassTransit/issues/581
17. https://masstransit.io/documentation/concepts/producers
18. https://www.reddit.com/r/csharp/comments/110e3jt/masstransit_using_default_rabbitmq_exchange/
19. https://www.youtube.com/watch?v=AOz1YPOKvEs
20. https://masstransit.io/documentation/configuration/transports/rabbitmq
21. https://www.rabbitmq.com/docs/troubleshooting-networking
22. https://www.freekb.net/Article?id=2962
23. https://www.freekb.net/Article?id=3019
24. https://groups.google.com/g/rabbitmq-users/c/tVK970gbMck
25. https://www.rabbitmq.com/docs/heartbeats
26. https://groups.google.com/g/rabbitmq-users/c/oYVfS5g4Jxc
27. https://habr.com/ru/articles/758938/
28. https://stackoverflow.com/questions/50698343/masstransit-request-response-sample
29. https://masstransit.io/documentation/configuration/transports/rabbitmq
30. https://github.com/rabbitmq/rabbitmq-dotnet-client/discussions/1213
31. https://stackoverflow.com/questions/66103468/why-do-code-changes-to-my-masstransit-application-cause-masstransit-rabbitmqtran
32. https://stackoverflow.com/questions/77610114/c-sharp-masstransit-consuming-messages-after-creates-new-queue-in-runtime-app
33. https://github.com/MassTransit/MassTransit/discussions/5785
34. https://github.com/rabbitmq/rabbitmq-dotnet-client/issues/1749
35. https://masstransit.io/support/common-mistakes
36. https://github.com/MassTransit/MassTransit/discussions/5782
37. https://groups.google.com/g/masstransit-discuss/c/WjL9haLOP3Y
38. https://masstransit.io/documentation/concepts/exceptions
39. https://masstransit.io/documentation/transports/rabbitmq
40. https://groups.google.com/g/masstransit-discuss/c/qgLPy_bW-uc

41. https://stackoverflow.com/questions/2148746/the-operator-is-reserved-for-future-use
42. https://www.reddit.com/r/C_Programming/comments/n8wvox/why_do_i_get_this_error_the_operator_is_reserved/
43. https://github.com/docker/for-win/issues/12548
44. https://forums.docker.com/t/out-of-box-fail-powershell/81233
45. https://github.com/PowerShell/vscode-powershell/issues/2780
46. https://forum.storj.io/t/i-got-error-operator-is-reserved-for-future-use/9782
47. https://stackoverflow.com/questions/51760214/how-to-restart-docker-for-windows-process-in-powershell
48. https://stackoverflow.com/questions/31365827/cannot-stop-or-restart-a-docker-container
49. https://suay.site/?p=3223
50. https://www.warp.dev/terminus/docker-restart-container
51. https://groups.google.com/g/masstransit-discuss/c/WjL9haLOP3Y
52. https://stackoverflow.com/questions/66103468/why-do-code-changes-to-my-masstransit-application-cause-masstransit-rabbitmqtran
53. https://github.com/MassTransit/MassTransit/discussions/5785
54. https://masstransit.io/documentation/concepts/exceptions
55. https://masstransit.io/support/upgrade
56. https://masstransit.io/documentation/configuration
57. https://github.com/MassTransit/MassTransit/issues/1752
58. https://stackoverflow.com/questions/68566100/servicebusconnectionexception-receive-transport-faulted
59. https://www.youtube.com/watch?v=-IpGYnznbco
60. https://github.com/MassTransit/MassTransit/issues/4372
61. https://masstransit.io/documentation/configuration/middleware/transactions