To Azure Service Bus and Beyond

# Azure Service Bus “by design”

For many right reasons message has size limitation. For example, large message has network latency problem with performance impact, large message has impact on network traffic, message receiver could have problem to manage large message, small is faster than larger and so on.

Azure service bus has many reasons to limit message size and the size limit is 256KB for “Standard tier” and 1MB for (Premium tier). Premium tier has other differences then Standard tier but the price is much higher.

You can read more details about quota limitations at this link:

<https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-quotas>

# Possible issues with those limitation

## Chatty messages

Frequently, to be sure to respect quota limit, the message contains only the identity of the object for which the event was raised and some information about the event.

In this situation who receive the message hasn’t sufficient information about the object and the only thing that can do is to recall a service that host the object and ask for information about the object using the identity contained in the event.

In this scenario when something happen to one object, this object send a message to service bus, and this message is consumed by event subscribers (subscribers can be many). At this point, every subscriber call the service’s object to ask information about it.

Therefore, for one message sent to service bus there are one message for each subscriber and one message from every subscriber to service‘s object.

It is a good practice to define a general purposed representation of the object represented by the message to reduce the “chatty” problem, for this, the message contains the most important information but not all the information or the complete object. In this way, only subscribers that needs information not contained in the message will recall the service’s object.

## Quota exceeded

Even if you have a message object that usually is smaller than 256 Kb, this object could exceed the size limitation in unexpected manner.

The object can have string properties that together exceed the limit.

The object can have an array of properties or an array of objects and the array has not size limit. In this scenario growing the size of the array, increase the probability to exceed the limit.

In those scenarios, you cannot be sure that your message will never exceed the size limitation.

## Expensive solution

You can adopt “Premium Tier” so you can exceed the 256Kb, anyway this tier has 1 Mb quota limitation.

“Premium Tier” has much performance than “Standard Tier” so is very expensive. I do not think this is the right solution for the message size problem. Anyway, the message’s size limit remain.

# The proposed solution

The proposed solution is applicable to all tier from “Basic” to “Standard” and “Premium”.

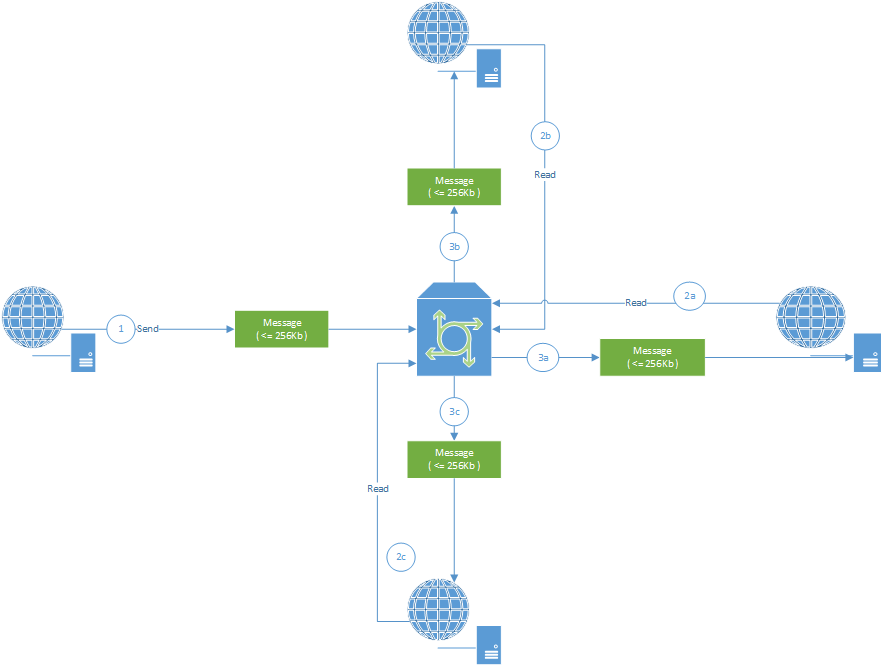
The library act as a wrapper of .net object used to send and receive message with Azure Service Bus.

The library expose a simplified façade to send and receive messages that exceeded the 256kb (or 1 Mb) size limit.

The library implement many strategies to send and receive large message. The strategy will be dynamically selected and will be used transparently.

## Strategy “foo”

The message is analyzed and if the size is lower than limitation is sent as is. The receiver accept the message without any rework on it.



The sender (at step 1) send the message.

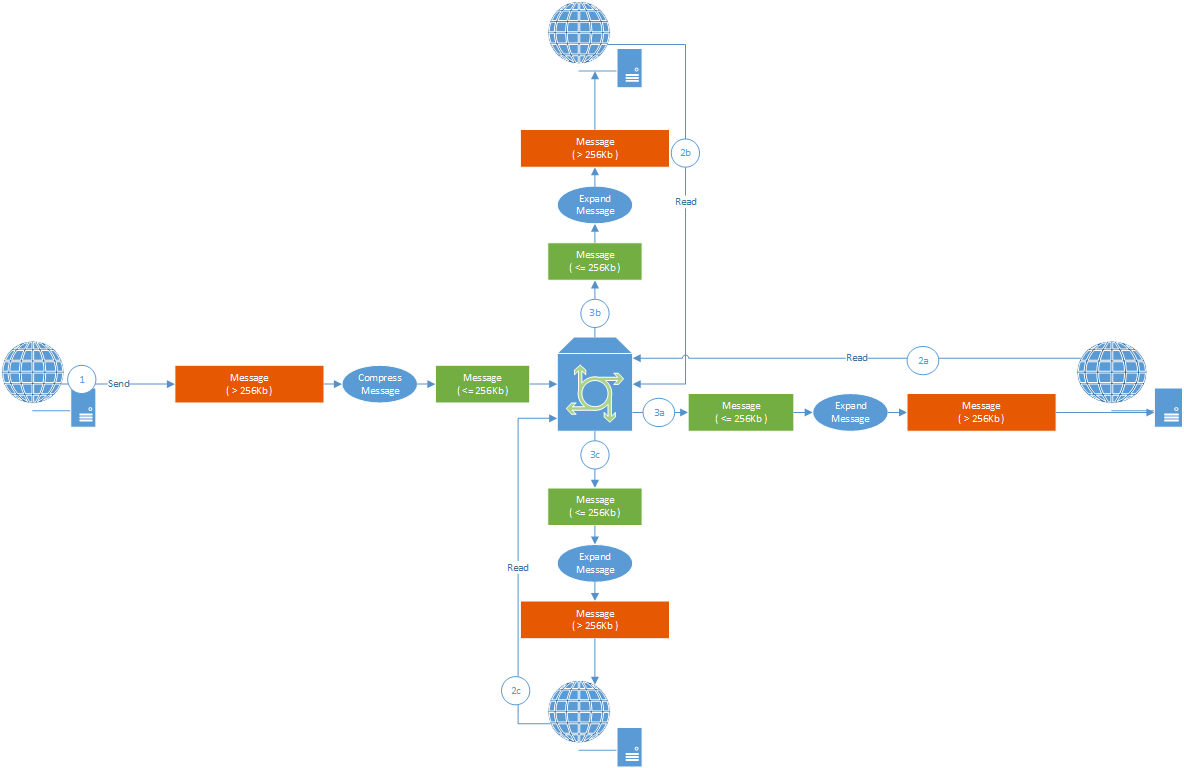
The library (green block) check the message size and if is lower than 256Kb sent it to Azure Service Bus.

Any receiver get the message form Azure Service Bus (step 2) using the library.

The library analyze the message (step 3) and understands that do not need any work on it and passes it to the receiver.

## Strategy “squeeze”

The message is analyzed and if the size is greater than limit, it will be compressed. The receiver understands that the message is compressed and decompress it before pass it to the receiver.



The sender (at step 1) send the message.

The library (red block) check the message’s size and if it is greater than 256 Kb, compress the message. After compression if the size is lower than 256Kb (or 1Mb) send the compressed message to Azure Service Bus.

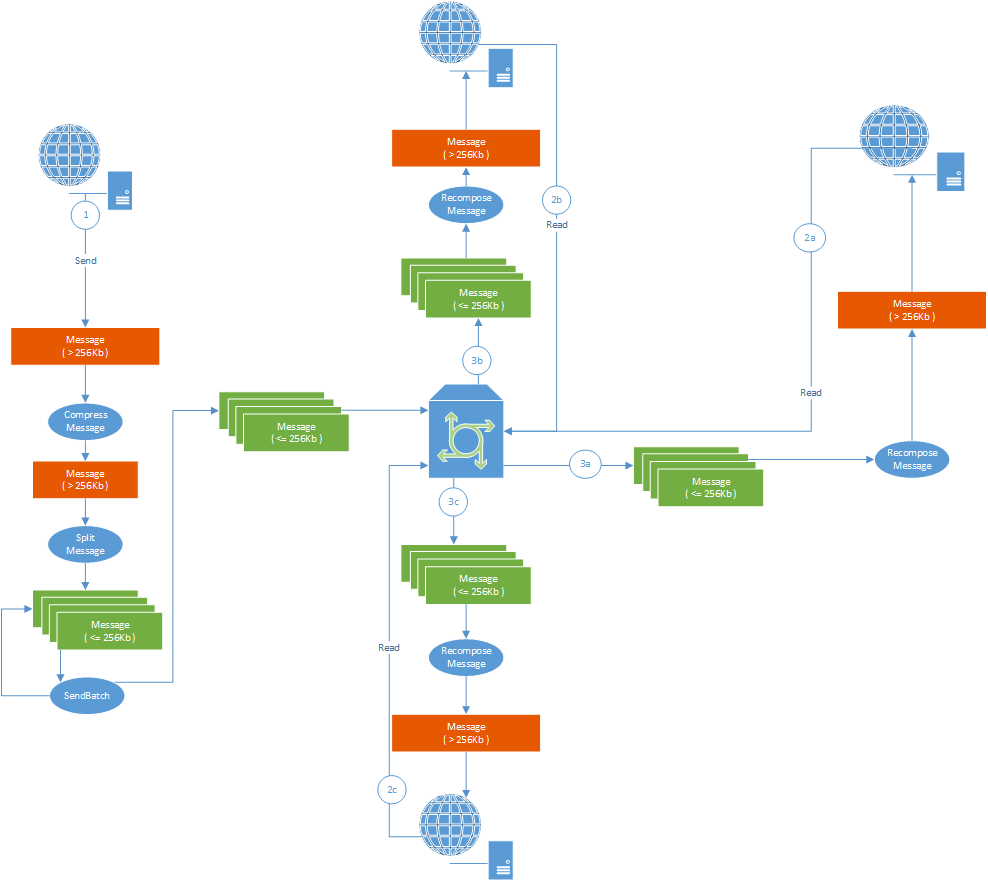
Any receiver get the message form Azure Service Bus (step 2) using the library.

The library analyze the message (step 3) and understands that is compressed so decompress it before passes it to the receiver.

If compression is not enough because the size is always greater than 256Kb after compression the library can use other strategies as “chunks” or “parking”.

## “Chunk” strategy

When compression is not enough, the library can split the compressed message in many message where every message has size lower then 256Kb. The library send all chunks in batch mode to Azure Service Bus.



The sender (at step 1) send the message.

The library (red block) check the message size and if it is greater than 256 Kb, compress the message.

If the size after compression is greater than 256Kb, the library split the message in many chunk with size lower than 256Kb and send them to Azure Service Bus in batch mode.

Any receiver get the message form Azure Service Bus (step 2) using the library.

The library analyze the message (step 3) and understands that is a part of compressed and splitted message. The library recompose message, decompress it and pass it to the receiver.

### Problem with this strategy

If a problem occurs while receiving a message with a chunk and the problem is not notified to Azure Service Bus, the latter does not retry to resend the message so the entire message will be corrupted and impossible to recover. This is a poor solution but require less Azure configurations.

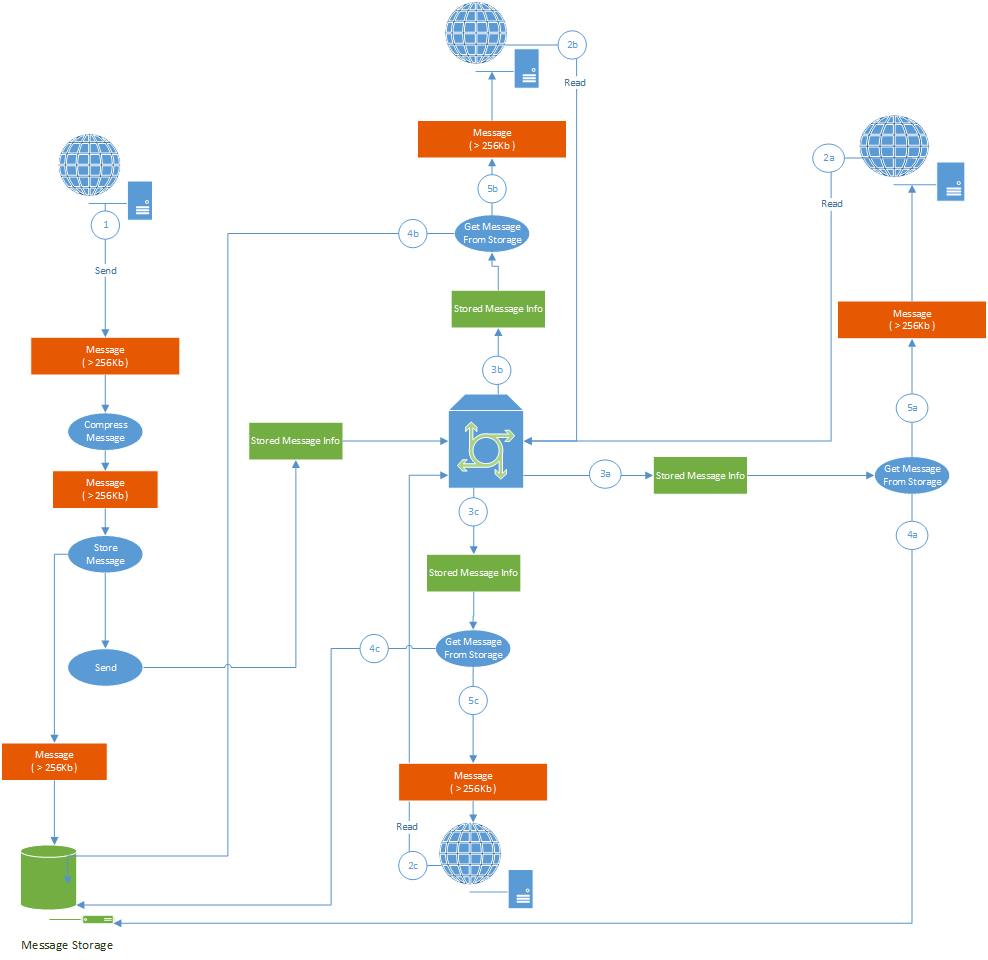
## “Parking” strategy

Parking strategy is a good alternative to “chunk” strategy.

The library can be configured to use “parking” strategy instead of “chunk” strategy.

If compressed message size is greater than 256Kb library can put the message in a blob storage.

At this time, the storage can be only a blob storage (today other ways like DocumentDB or MongoDB are not implemented).



The sender (at step 1) send the message.

The library (red block) check the message size and if it is greater than 256 Kb, compress the message.

If the size after compression is greater than 256Kb the library put the message into blob storage and send a “stub” message to Azure Service Bus.

Any receiver get the message form Azure Service Bus (step 2) using the library.

The library analyze the message (step 3) and understands that is only a stub with information about original message. Then the library get the message from the storage, decompress it and pass it to the receiver.