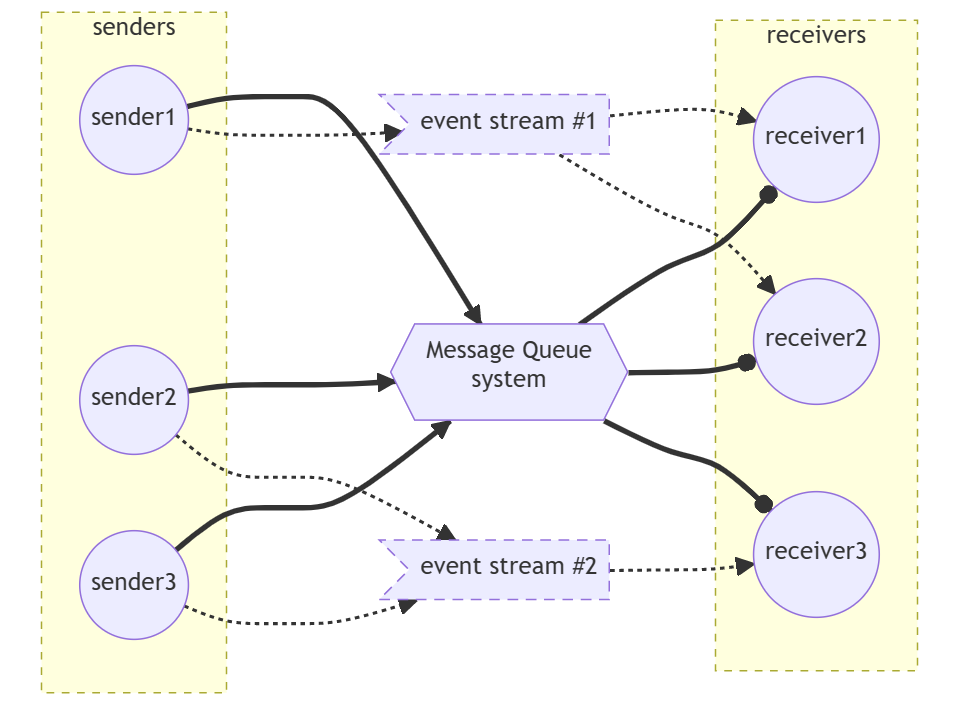
# Orleans的Grain事件發送機制：Stream事件流

Orleans的Grain除了昨天講述的Observer Pattern樣式的事件通知功能之外，事件流（event stream，以下簡稱Stream），是個能提供『穩定』的Grain與Grain、Grain與外界物件之間非同步、響應式執行動作或傳送訊息的機制。

## Orleans Stream簡介

Stream的機制類似訊息佇列(Message Queue)，在Grain實作開發時，開發的思維模型可想像為底層有個事件流的機制，讓開發者能夠在Grain之間、Grain與外界物件之間，透過事件流的機制，來進行非同步、響應式的執行動作或傳送訊息：



如上圖，Sender和Receiver之間藉由訊息佇列(Message Queue)來進行溝通，Sender將訊息送入Message Queue，Receiver則從Message Queue中取出訊息來進行處理。

在Orleans的Grain實作程式碼內，可在發送和接受Stream API中的 streamNamespace 的字串參數值，來邏輯區分為不同的事件流(event stream)來分類訊息，且一個Grain可發送/訂閱多個事件流，以便實現 Grain actor 之間的多對多溝通。

在運營(Ops)的Silo配置方面，需要配置『Stream Provider』，用來串接提供事件流儲存與發送機制的實際訊息佇列系統，Orleans官方提供的Stream Provider選擇方案有：

* [Memory Stream](https://learn.microsoft.com/dotnet/api/orleans.hosting.silobuildermemorystreamextensions.addmemorystreams)： 提供測試專案用的In-Memory Stream Provider。
* [Azure Queue Storage](https://azure.microsoft.com/products/storage/queues)： [Microsoft.Orleans.Streaming.AzureStorage](https://www.nuget.org/packages/Microsoft.Orleans.Streaming.AzureStorage) NuGet套件。
* [Azure Event Hub](https://azure.microsoft.com/products/event-hubs)： 在 v3.x 時要用[Microsoft.Orleans.OrleansServiceBus](https://www.nuget.org/packages/Microsoft.Orleans.OrleansServiceBus) NuGet套件，在 v7 以後正名為 [Microsoft.Orleans.Streaming.EventHubs](https://www.nuget.org/packages/Microsoft.Orleans.Streaming.EventHubs) NuGet套件。
* [AWS SQS(Amazon Simple Queue Service)](https://aws.amazon.com/sqs)： [Microsoft.Orleans.Streaming.SQS](https://www.nuget.org/packages/Microsoft.Orleans.Streaming.SQS) NuGet套件。

其他第三方的開源Stream Provider還有[Azure CosmosDB change feed](https://github.com/OrleansContrib/Orleans.CosmosDB)、[Kafka](https://github.com/OrleansContrib/Orleans.Streams.Kafka)、[RabbitMQ](https://github.com/KevinCathcart/Orleans.Streams.RabbitMqStreamProvider)、[Redis](https://github.com/berdon/Orleans.Providers.Redis)等等。

除了由Grain主動發送事件流的訊息之外，有些如Azure Event Hub的Stream Provider，可以撰寫自製DataAdapter，然後在Silo配置時用 [.UseDataAdapter()](https://learn.microsoft.com/dotnet/api/orleans.hosting.eventhubstreamconfiguratorextensions.usedataadapter) 來註冊該DataAdapter，將外部的事件流透過DataAdapter轉換成Orleans的事件流，使Grain能接收外部事件流資料，進行處理。

## Orleans Stream在Grain實作層的開發概念

1. Stream 只需要有識別子去呼叫API就可以取得，類似取得Grain的RPC呼叫參考一樣，不必顧慮是否需要開啟或關閉事件流的生命週期。
2. Stream的ID由Guid以及一個namespace字串組成，同樣的ID就能取得同樣的Stream，所以可做到多個Grain發送源共用同一個Stream將資料匯出，這是使用Observer的事件機制辦不到的事。
3. Stream設計成要使用時可快速連接，不使用時快速斷開，因此不需要一直保存著stream的參考，只要有ID就可以取得。
4. Stream的生命週期由Stream Provider來管理，在Grain實作層，只需要呼叫 [.GetStreamProvider()](https://learn.microsoft.com/dotnet/api/orleans.grain.getstreamprovider) 來取得Stream Provider的參考，然後再呼叫Stream Provider提供的 GetStream() 來取得Stream的物件實體參考，發送者Grain就可以開始發送事件流訊息了。
5. 發送者在取得Stream的參考後，可以多次呼叫 [.OnNextAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.iasyncobserver-1.onnextasync) 來發送一筆筆訊息出去到事件流，等到全部發送完畢，不再需要Stream時，呼叫 [.OnCompletedAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.iasyncobserver-1.oncompletedasync) 來宣告結束全部的發送階段，或是呼叫 [.OnErrorAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.iasyncobserver-1.onerrorasync) 當發送端有錯誤發生需強制結束時。  
   （不過有些Stream Provider提供的stream，例如In-Memory Stream，不需自行呼叫 OnCompleteAsync() & OnErrorAsync()這兩個API，只需將該stream物件參考設為null讓GC記憶體回收即可）
6. 訂閱者Grain要訂閱Stream的API，分為顯式和隱式兩種訂閱方法：
   * 顯式訂閱：取得stream物件參考之後，呼叫該stream的 [SubscribeAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.asyncobservableextensions.subscribeasync) 擴充方法來訂閱事件流，該API需要傳入一個實作 [IAsyncObserver<T>](https://learn.microsoft.com/dotnet/api/orleans.streams.iasyncobserver-1) 介面的物件，或是有實作該介面的 [OnNextAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.iasyncobserver-1.onnextasync) 型態的方法/Action/Lambda表示式，在裡面撰寫訂閱者Grain接收訊息的處理邏輯。  
     而呼叫 [SubscribeAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.asyncobservableextensions.subscribeasync) 之後取得的 StreamSubscriptionHandle<T> 型態的訂閱處理(handle)物件，不是完全沒有用途的：當Grain本身因為Silo故障等問題而被重新啟動時，需呼叫該訂閱處理物件之 [ResumeAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.streamsubscriptionhandle-1.resumeasync) 來重新訂閱事件流，並且在訂閱者Grain不再需要事件流訊息時，呼叫 [.UnsubscribeAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.iasyncobserver-1.unsubscribeasync) 來取消訂閱。
   * 隱式訂閱：在Grain實作類別上加掛 [ImplicitStreamSubscriptionAttribute](https://learn.microsoft.com/dotnet/api/orleans.implicitstreamsubscriptionattribute) 屬性，並實作 [IStreamSubscriptionObserver](https://learn.microsoft.com/dotnet/api/orleans.streams.core.istreamsubscriptionobserver) 介面的 [OnSubscribed()](https://learn.microsoft.com/dotnet/api/orleans.streams.core.istreamsubscriptionobserver.onsubscribed) 方法。  
     此種訂閱方式有兩個好處：
     1. Grain實體自動在訊息送達時自動觸發，不必手動呼叫其他特別設計的RPC方法讓Grain實體啟動。
     2. 不必在Grain本身因為Silo故障等問題而被重新啟動時，得呼叫 [UnsubscribeAsync()](https://learn.microsoft.com/dotnet/api/orleans.streams.iasyncobserver-1.unsubscribeasync) 這個恢復訂閱的API，Orleans Runtime會自動處理好這些問題。
7. 除了Grain實體之間能夠發送/訂閱事件流事件之外，Orleans RPC客戶端也可使用事件流的API來訂閱Grain的事件流事件。

至於Orleans Stream 在Silo的運營配置方面要怎麼做，明天以實作Orleans Steam的範例專案的方式來展示使用Azure Queue Storage作為Stream Provider。