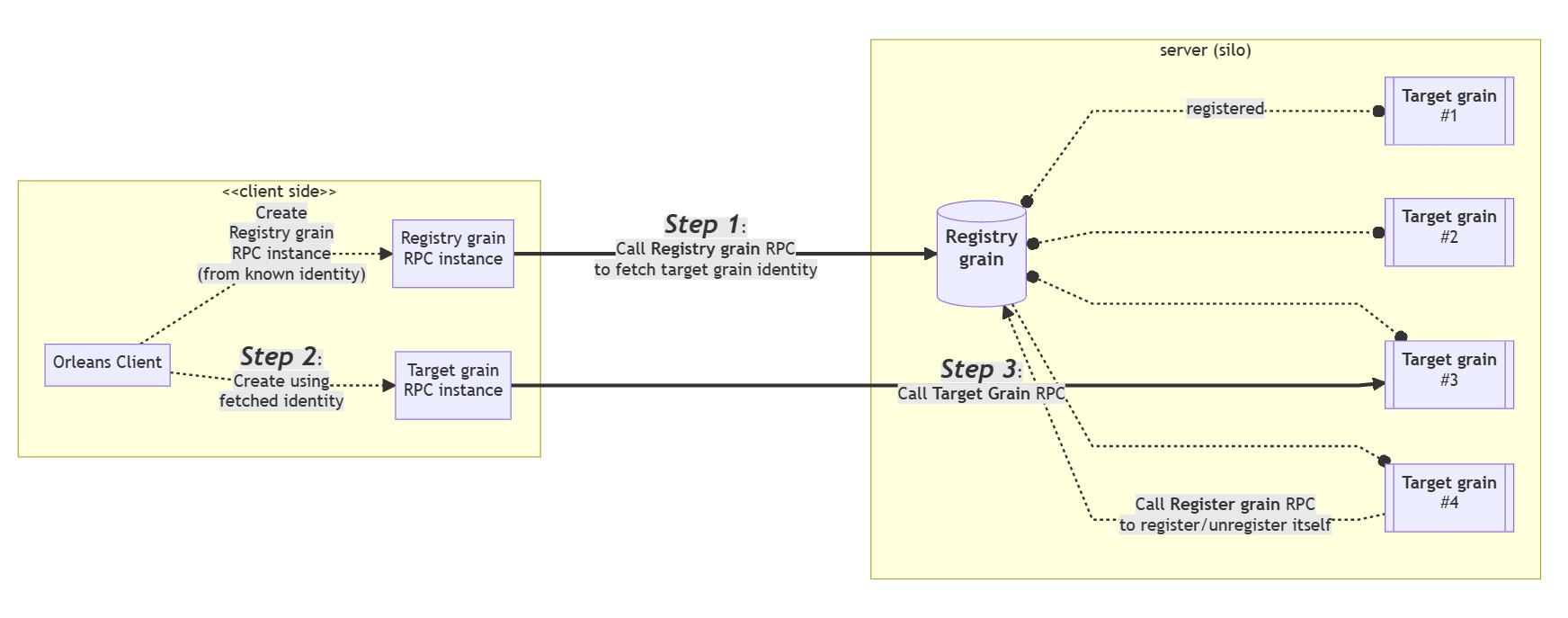
# Orleans常見系統架構模式：Registry Pattern及其應用範例

## Registry Pattern 介紹

[Registry Pattern](https://social.technet.microsoft.com/wiki/contents/articles/51018.registry-pattern-using-microsoft-orleans.aspx) 是Orleans一種用來解決無法得知想要與之互動的目標Grain個體，是否已經有被正確的初始化過而能放心呼叫商業邏輯RPC之設計模式，它的運作概念是將系統中已啟用過的Grain實體的識別子(identity)，登記到一或多個 RegistryGrain 中，並且由 RegistryGrain 來記錄/管理這些Grain資源，讓Client/RPC呼叫端可以透過這種 RegistryGrain 來取得那些Grain資源，以便進行後續的互動。

整個流程如下：



在一開始，Client端只知道用來建立RegistryGrain RPC呼叫參考的識別子字串、數字、GUID值等的資訊，所以就先建立出RPC呼叫參考，再用呼叫它的RPC方法，取得真正要互動的目標Grain識別子(identity)，接著Client端就可以透過這個取得的識別子來建立目標Grain的RPC實體，並且進行後續的互動。

採用Registry Pattern的目的，如同Microservice技術領域中的 [Service registry pattern](https://microservices.io/patterns/service-registry.html) 一樣，都是為了要解決服務後端眾多資源有可能因為動態變動位址/狀態等前端無法掌控的因素，導致前端不知道實際情形而無從使用的問題；之前有個可以取代此Pattern的實驗性專案（[Orleans.Indexing](https://github.com/OrleansContrib/Orleans.Indexing)），可在Silo內部的背景服務建立和維護Grain的查詢索引，以便讓Client端用類似Linq查詢的方式來直接尋找並定位想要的Grain參考，後續進行RPC呼叫互動，不過目前此專案是處於是開發暫停的狀態。

Registry Pattern在使用上需注意的是，Registry grain本身有可能會成為單點故障的根源，而且當單一個Registry Grain管理了太多個target grain時，Registry grain本身的負載也會變得非常大，形成效能瓶頸。

Registry Pattern可用在如購物車、目前游戲在線玩家列表等功能實現。

## Registry Pattern 應用範例