FlexMatch 開發人員指南

Amazon GameLift Servers

版本

Amazon GameLift Servers: FlexMatch 開發人員指南

Table of Contents

什麼是Amazon GameLift ServersFlexMatch？

Amazon GameLift Servers FlexMatch 是多玩家遊戲的可自訂配對服務。使用 FlexMatch，您可以建立一組自訂規則，定義多玩家配對在遊戲中的樣子，並決定如何評估和選取每個配對的相容玩家。您也可以微調配對演算法的關鍵層面，以符合您的遊戲需求。

使用 FlexMatch做為獨立配對服務，或與Amazon GameLift Servers遊戲託管解決方案整合。例如，您可以使用具有peer-to-peer架構的遊戲或使用其他雲端運算解決方案的遊戲，實作 FlexMatch 做為獨立功能。或者，您可以將 FlexMatch 新增至Amazon GameLift Servers受管 EC2 或受管容器託管，或使用 Amazon GameLift ServersAnywhere 的內部部署託管。本指南提供如何為特定案例建置FlexMatch配對系統的詳細資訊。

FlexMatch 可讓您根據遊戲需求靈活地設定配對優先順序。例如，您可以執行下列動作：

- 尋找配對速度和品質之間的平衡。設定配對規則以快速尋找夠好的配對，或讓玩家等待更長的時間，以找到最佳配對來獲得最佳玩家體驗。
- 根據配對良好的玩家或配對良好的隊伍進行配對。建立所有玩家具有類似特性的配對，例如技能或經驗。或表單比對，其中每個團隊的合併特性符合共同條件。
- 排定玩家延遲如何納入配對的優先順序。您想要為所有玩家設定硬性延遲限制，還是只要配對中的每個人都有類似的延遲，就可以接受較高的延遲？

> **準備好開始使用 FlexMatch了嗎？**
>
> 如需使用 啟動和執行遊戲的step-by-step指引FlexMatch，請參閱下列主題：
>
> - 路線圖：將配對新增至Amazon GameLift Servers託管解決方案
> - 路線圖：使用 建立獨立的配對解決方案 FlexMatch

主要FlexMatch功能

無論您使用 FlexMatch做為獨立服務或搭配Amazon GameLift Servers遊戲託管，下列功能適用於所有FlexMatch案例。

- 可自訂玩家配對。設計和建置配對建構器，以符合您提供玩家的所有遊戲模式。建置一組自訂規則來評估關鍵玩家屬性（例如技能等級或角色）和地理延遲資料，為您的遊戲形成絕佳的玩家配對。

- 以延遲為基礎的比對。提供玩家延遲資料並建立配對規則，要求配對中的玩家有類似的回應時間。當您的玩家配對集區跨越多個地理區域時，此功能非常有用。
- 支援最多 200 名玩家的配對大小。使用為您的遊戲自訂的配對規則，建立最多 40 名玩家的配對。使用使用簡化的自訂配對程序來建立最多 200 名玩家的配對，讓玩家等待時間可以管理。
- 玩家接受。在完成配對並開始遊戲工作階段之前，要求玩家選擇加入提議的配對。使用此功能啟動您的自訂接受工作流程，並在為配對放置新的遊戲工作階段FlexMatch之前向 報告玩家回應。如果並非所有玩家都接受配對，則提議的配對會失敗，而接受的玩家會自動返回配對集區。
- 玩家方支援。為想要在同一隊伍一起玩的玩家群組產生配對。使用 FlexMatch 尋找其他玩家，視需要填寫配對。
- 可擴展的相符規則。在經過一定的時間後逐漸放寬配對要求，而不會找到成功的配對。規則擴展可讓您決定放寬初始配對規則的位置和時間，讓玩家可以更快進入可玩遊戲。
- 符合回填。將現有遊戲工作階段中的空玩家位置填入配對良好的新玩家。自訂請求新玩家的時間和方式，並使用相同的自訂配對規則來尋找其他玩家。

FlexMatch 使用 Amazon GameLift Servers 託管

FlexMatch 提供下列其他功能，可搭配您透過 託管的遊戲使用Amazon GameLift Servers。這包括具有自訂遊戲伺服器或 的遊戲Amazon GameLift ServersRealtime。

- 遊戲工作階段置放。當成功配對時， FlexMatch會自動向 請求新的遊戲工作階段放置Amazon GameLift Servers。在配對過程中產生的資料，包括玩家 IDs和團隊指派，會提供給遊戲伺服器，以便其可以使用該資訊來啟動配對的遊戲工作階段。FlexMatch然後， 會傳回遊戲工作階段連線資訊，讓遊戲用戶端可以加入遊戲。為了將玩家在配對中遇到的延遲降至最低，使用 的遊戲工作階段置放Amazon GameLift Servers也可以使用區域玩家延遲資料。
- 自動配對回填。啟用此功能後，當新的遊戲工作階段從未填滿的玩家位置開始時， FlexMatch會自動傳送配對回填請求。您的配對系統會以最少的玩家數量開始遊戲工作階段置放程序，然後快速填滿剩餘的位置。您無法使用自動回填來取代退出相符遊戲工作階段的玩家。

如果您使用 Amazon GameLift ServersFleetIQ 搭配 Amazon Elastic Compute Cloud (Amazon EC2) 資源託管的遊戲，請將 實作FlexMatch為獨立服務。

的定價 Amazon GameLift ServersFlexMatch

Amazon GameLift Servers 依使用持續時間計算的執行個體費用，以及依傳輸資料數量計算的頻寬費用。如果您在 上託管遊戲Amazon GameLift Servers，FlexMatch用量會包含在 的費用中Amazon

GameLift Servers。如果您在另一個伺服器解決方案上託管遊戲，則會另外收取FlexMatch使用費。如需 Amazon GameLift Servers 的完整收費清單與定價，請參閱 Amazon GameLift Servers 定價。

如需使用 計算託管遊戲或配對成本的資訊Amazon GameLift Servers，請參閱產生Amazon GameLift Servers定價預估，其中說明如何使用 AWS 定價計算工具。

如何Amazon GameLift Servers FlexMatch運作

本主題提供 Amazon GameLift ServersFlexMatch服務的概觀，包括FlexMatch系統的核心元件及其互動方式。

您可以FlexMatch搭配使用Amazon GameLift Servers受管託管的遊戲使用 ，或是搭配使用其他託管解決方案的遊戲使用 。託管在 上的遊戲Amazon GameLift Servers，包括Amazon GameLift Servers Realtime，使用整合Amazon GameLift Servers服務來自動尋找可用的遊戲伺服器，並為配對啟動遊戲工作階段。使用 FlexMatch做為獨立服務的遊戲，包括 Amazon GameLift Servers FleetIQ，必須與現有的託管系統協調，為配對指派託管資源並啟動遊戲工作階段。

如需FlexMatch設定遊戲的詳細指引，請參閱 FlexMatch 入門。

配對元件

FlexMatch 配對系統包含以下部分或全部元件。

Amazon GameLift Servers 元件

這些Amazon GameLift Servers資源可控制FlexMatch服務如何為您的遊戲執行配對。它們是使用Amazon GameLift Servers工具建立和維護，包括 主控台和 AWS CLI，或者，以程式設計方式使用適用於 的 AWS SDKAmazon GameLift Servers。

- FlexMatch 配對組態 （ 也稱為配對建構器 ） – 配對建構器是一組組態值，可自訂遊戲的配對程序。遊戲可以有多個配對建構器，每個配對建構器會視需要針對不同的遊戲模式或體驗進行設定。當您的遊戲將配對請求傳送至 時FlexMatch，它會指定要使用的配對建構器。
- FlexMatch 配對規則集 – 規則集包含評估玩家潛在配對以及核准或拒絕所需的所有資訊。規則集會定義配對的團隊結構、宣告用於評估的玩家屬性，並提供描述可接受配對條件的規則。規則可以套用到個別玩家、隊伍或整個配對。例如，規則可能需要配對中的每個玩家都選擇相同的遊戲地圖，或者可能需要所有隊伍都有類似的玩家技能平均值。
- Amazon GameLift Servers 遊戲工作階段佇列 （ 僅限FlexMatch具有Amazon GameLift Servers受管託管的) – 遊戲工作階段佇列會尋找可用的託管資源，並為配對啟動新的遊戲工作階段。佇列的組態會決定在何處Amazon GameLift Servers尋找可用的託管資源，以及如何為配對選取最佳的可用主機。

自訂元件

下列元件包含完整FlexMatch系統所需的功能，您必須根據遊戲的架構實作這些功能。

- 配對的玩家界面 – 此界面可讓玩家加入配對。它至少會透過用戶端配對服務元件啟動配對請求，並視需要提供玩家特定的資料，例如技能水準和延遲資料。

  > **Note**
  >
  > 最佳實務是，與 FlexMatch服務的通訊應該由後端服務完成，而不是從遊戲用戶端完成。

- 用戶端配對服務 – 此服務會從玩家界面將玩家聯結請求建立欄位、產生配對請求，並將請求傳送至FlexMatch服務。對於處理中的請求，它會監控配對事件、追蹤配對狀態，並視需要採取行動。視您在遊戲中管理遊戲工作階段託管的方式而定，此服務可能會將遊戲工作階段連線資訊傳回給玩家。此元件使用 AWS SDK 搭配 Amazon GameLift Servers API 來與服務通訊FlexMatch。
- 配對置放服務 （僅適用於 FlexMatch做為獨立服務） – 此元件可與現有的遊戲託管系統搭配使用，以尋找可用的託管資源，並為配對啟動新的遊戲工作階段。元件必須取得配對結果，並擷取啟動新遊戲工作階段所需的資訊，包括配對中所有玩家的玩家 IDs、屬性和隊伍指派。

FlexMatch 配對程序

本主題說明基本配對案例中的事件順序，包括各種遊戲元件與 FlexMatch服務之間的互動。

步驟 1：為玩家請求配對

使用遊戲用戶端的玩家按一下「加入遊戲」按鈕。此動作會導致您的用戶端配對服務將配對請求傳送至 FlexMatch。請求會識別滿足請求時要使用的FlexMatch配對建構器。請求也包含自訂配對建構器所需的玩家資訊，例如技能水準、播放偏好設定或地理延遲資料。您可以對一個玩家或多個玩家提出配對請求。

步驟 2：將請求新增至配對集區

當 FlexMatch收到配對請求時，會產生配對票證，並將其新增至配對建構器的票證集區。票證會保留在集區中，直到相符或達到時間上限為止。您的用戶端配對服務會定期收到配對事件的通知，包括票證狀態的變更。

步驟 3：建立配對

您的FlexMatch配對建構器會持續在其集區中的所有票證上執行下列程序：

1. 配對建構器會依票證存留期排序集區，然後開始從最舊的票證開始建置潛在的配對。

2. 配對建構器會將第二個票證新增至潛在配對，並根據自訂配對規則評估結果。如果潛在配對通過評估，票證的玩家會指派給團隊。
3. 配對建構器會依序新增下一個票證，並重複評估程序。當所有玩家位置都填滿後，配對就會準備就緒。

大型配對的配對 (41 到 200 名玩家） 使用上述程序的修改版本，以便在合理的時間範圍內建置配對。配對建構器不會個別評估每個票證，而是將預先排序的票證集區分成潛在配對，然後根據您指定的玩家特性來平衡每個配對。例如，配對建構器可能會根據類似的低延遲位置預先排序票證，然後使用配對後平衡來確保隊伍平均地符合玩家技能。

步驟 4：報告配對結果

找到可接受的相符項目時，所有相符的票證都會更新，並為每個相符的票證產生成功的配對事件。

- FlexMatch 作為獨立服務：您的遊戲會收到成功配對事件的配對結果。結果資料包含所有配對玩家及其團隊指派的清單。如果您的配對請求包含玩家延遲資訊，結果也會建議配對的最佳地理位置。
- FlexMatch 使用Amazon GameLift Servers託管解決方案：比對結果會自動傳遞至 Amazon GameLift Servers佇列以進行遊戲工作階段放置。配對建構器會決定用於遊戲工作階段置放的佇列。

步驟 5：啟動配對的遊戲工作階段

成功形成提議的配對後，就會啟動新的遊戲工作階段。設定配對的遊戲工作階段時，您的遊戲伺服器必須能夠使用配對結果資料，包括玩家 IDs 和團隊指派。

- FlexMatch 作為獨立服務：您的自訂配對置放服務會從成功的配對事件取得配對結果資料，並連接到現有的遊戲工作階段置放系統，以尋找配對的可用託管資源。找到託管資源後，配對置放服務會與您現有的託管系統協調，以啟動新的遊戲工作階段並取得連線資訊。
- FlexMatch 使用Amazon GameLift Servers託管解決方案：遊戲工作階段佇列會尋找配對的最佳可用遊戲伺服器。根據佇列的設定方式，它會嘗試將遊戲工作階段放置在成本最低的資源，以及玩家會遇到低延遲的位置 （如果提供玩家延遲資料）。遊戲工作階段成功放置後，Amazon GameLift Servers服務會提示遊戲伺服器啟動新的遊戲工作階段，傳遞配對結果和其他選用的遊戲資料。

步驟 6：將玩家連接到配對

遊戲工作階段開始後，玩家會連線至工作階段、宣告其團隊指派，並開始遊戲。

- FlexMatch 作為獨立服務：您的遊戲使用現有的遊戲工作階段管理系統，將連線資訊傳回給玩家。

- FlexMatch 使用Amazon GameLift Servers託管解決方案：在成功的遊戲工作階段放置上， 會使用遊戲工作階段連線資訊和玩家工作階段 ID FlexMatch更新所有相符的票證。

FlexMatch 支援 AWS 區域

如果您使用 FlexMatch搭配 Amazon GameLift Servers託管解決方案，您可以在託管遊戲的任何位置託管相符的遊戲工作階段。請參閱 AWS 區域 和 Amazon GameLift Servers 託管位置的完整清單。

FlexMatch 入門

使用本節中的資源，協助您開始使用 建置配對系統FlexMatch。

主題

- 設定 AWS 帳戶 的 FlexMatch
- 路線圖：使用 建立獨立的配對解決方案 FlexMatch
- 路線圖：將配對新增至Amazon GameLift Servers託管解決方案

設定 AWS 帳戶 的 FlexMatch

Amazon GameLift Servers FlexMatch 是一項 AWS 服務，您必須擁有 AWS 帳戶才能使用此服務。建立 AWS 帳戶是免費的。如需如何使用 AWS 帳戶的詳細資訊，請參閱 入門 AWS。

如果您使用 FlexMatch 搭配其他Amazon GameLift Servers解決方案，請參閱下列主題：

- 設定Amazon GameLift Servers託管的存取權
- 設定使用 託管的存取權 Amazon GameLift ServersFleetIQ

為 設定您的帳戶 Amazon GameLift Servers

1. 取得帳戶。開啟 Amazon Web Services，然後選擇登入主控台。依照提示建立新的帳戶或登入現有的帳戶。
2. 設定管理使用者群組。開啟 AWS Identity and Access Management (IAM) 服務主控台，並依照步驟建立或更新使用者或使用者群組。IAM 會管理對 AWS 服務和資源的存取。每個使用 Amazon GameLift Servers主控台或呼叫 Amazon GameLift Servers APIs 存取 FlexMatch 資源的人，都必須獲得明確存取權。如需使用主控台（或 AWS CLI 或其他工具）設定使用者群組的詳細說明，請參閱建立 IAM 使用者。
3. 將許可政策連接至您的使用者或使用者群組。 將 IAM 政策連接至使用者或使用者群組，以管理對 AWS 服務和資源的存取。許可政策會指定一組使用者可存取 AWS 的服務和動作。

   對於 Amazon GameLift Servers，您必須建立自訂許可政策，並將其連接到每個使用者或使用者群組。政策是 JSON 文件。使用以下範例來建立您的政策。

下列範例說明具有所有Amazon GameLift Servers資源和動作管理許可的內嵌許可政策。您可以選擇只指定 FlexMatch特定項目來限制存取。

JSON

```
{
"Version":"2012-10-17",
"Statement":
  {
    "Effect": "Allow",
    "Action": "gamelift:*",
    "Resource": "*"
  }
}
```

路線圖：使用 建立獨立的配對解決方案 FlexMatch

本主題概述將 實作FlexMatch為獨立配對服務的完整整合程序。如果您的多玩家遊戲是使用peer-to-peer、自訂設定的現場部署硬體或其他雲端運算基本概念託管，請使用此程序。此程序也適用於Amazon GameLift ServersFleetIQ，這是 Amazon EC2 上託管遊戲的託管最佳化解決方案。如果您使用Amazon GameLift Servers受管託管 （ 包括 Amazon GameLift ServersRealtime) 託管遊戲，請參閱路線圖：將配對新增至Amazon GameLift Servers託管解決方案。

開始整合之前，您必須擁有 AWS 帳戶並設定Amazon GameLift Servers服務的存取許可。如需詳細資訊，請參閱設定 AWS 帳戶 的 FlexMatch。與建立和管理Amazon GameLift ServersFlexMatch配對建構器和規則集相關的所有基本任務都可以使用 Amazon GameLift Servers主控台完成。

1. 建立FlexMatch配對規則集。您的自訂規則集提供如何建構相符項目的完整指示。您可以在其中定義每個團隊的結構和大小。您也會提供一組要求，配對必須符合才能有效，這會FlexMatch使用 在配對中包含或排除玩家。這些要求可能適用於個別玩家。您也可以自訂規則集中的FlexMatch演算法，例如建立最多 200 名玩家的大型配對。請參閱以下主題：

   - 建置FlexMatch規則集
   - FlexMatch 規則集範例

2. 設定配對事件的通知。使用通知來追蹤FlexMatch配對活動，包括待定配對請求的狀態。這是用來交付提議配對結果的機制。配對請求並非同步，因此需要追蹤請求狀態的方法。使用通知是此選項的偏好選項。請參閱以下主題：

- 設定FlexMatch事件通知
- FlexMatch 配對事件

3. 設定FlexMatch配對組態。也稱為配對建構器，此元件會接收配對請求並進行處理。您可以透過指定規則集、通知目標和最長等待時間來設定配對建構器。您也可以啟用選用功能。請參閱以下主題：

   - 設計FlexMatch配對建構器
   - 建立配對組態

4. 建置用戶端配對服務。建立或擴展遊戲用戶端服務，具有建置和傳送配對請求至 的功能 FlexMatch。若要建置配對請求，此元件必須具有機制，以取得配對規則集所需的玩家資料，以及選擇性的區域延遲資訊。它還必須具有為每個請求建立和指派唯一票證 IDs的方法。您也可以選擇建置玩家接受工作流程，要求玩家選擇加入提議的配對。此服務也必須監控配對事件，以取得配對結果，並為成功配對啟動遊戲工作階段放置。請參閱此主題：

   - FlexMatch 新增至遊戲用戶端

5. 建置配對置放服務。 建立可與現有遊戲託管系統搭配使用的機制，以尋找可用的託管資源，並為成功配對啟動新的遊戲工作階段。此元件必須能夠使用配對結果資訊來取得可用的遊戲伺服器，並為配對啟動新的遊戲工作階段。您可能也想要實作工作流程來提出配對回填請求，這會使用配對來填滿已在執行的配對遊戲工作階段中開啟的空位。

路線圖：將配對新增至Amazon GameLift Servers託管解決方案

FlexMatch 適用於自訂遊戲伺服器和 的 受管Amazon GameLift Servers託管Amazon GameLift ServersRealtime。若要在遊戲中加入 FlexMatch 配對功能，請完成以下任務。

- 設定配對建構器。配對建構器會收到玩家的配對請求，並加以處理。其會根據定義好的規則將玩家分組，每成功配對一組，便建立新的遊戲工作階段和玩家工作階段。請按照以下步驟設定配對建構器：
  - 建立規則集。規則集會告訴配對建構器如何建構有效的配對。其會指定隊伍的結構，並指定如何評估玩家加入配對的資格。請參閱以下主題：
    - 建置FlexMatch規則集
    - FlexMatch 規則集範例
  - 建立遊戲工作階段佇列。佇列會找出最適合各個配對的區域，並在該區域內建立新的遊戲工作階段。使用既有的佇列，或建立新佇列以供進行配對。請參閱此主題：
    - 建立佇列

  - 設定通知 (選用)。配對請求並非同步，因此需要追蹤請求狀態的方法。建議使用通知。請參閱此主題：
    - 設定FlexMatch事件通知
  - 設定配對建構器組態。擁有規則集、佇列、通知目標後，請建立配對建構器的組態。請參閱以下主題：
    - 設計FlexMatch配對建構器
    - 建立配對組態
- 將 FlexMatch 整合至遊戲用戶端服務。將功能新增到遊戲用戶端服務，以利用配對開始新遊戲工作階段。配對請求會指定哪些配對建構器會使用並為配對提供必要的玩家資料。請參閱此主題：
  - FlexMatch 新增至遊戲用戶端
- 將 FlexMatch 整合至遊戲伺服器。將功能新增到遊戲伺服器，以便透過配對開始遊戲工作階段。此類型遊戲工作階段的請求包含配對特定資訊，其中包含玩家及團隊指派。針對配對建構遊戲工作階段時，遊戲伺服器需要存取並使用此資訊。請參閱此主題：
  - FlexMatch 新增至 Amazon GameLift Servers託管的遊戲伺服器
- 設定 FlexMatch 回填功能 (選用)。要求額外的玩家配對以填補目前遊戲中空缺的玩家位置。您可以開啟自動回填，讓 Amazon GameLift Servers 管理回填請求。或者，您可以藉由將功能新增到遊戲用戶端服務或遊戲伺服器，以手動方式管理回填，進而啟動配對回填請求。請參閱此主題：
  - 使用 回填現有遊戲 FlexMatch

> **Note**
>
> FlexMatch 回填目前不適用於使用 的遊戲Amazon GameLift ServersRealtime。

綁定Amazon GameLift Servers FlexMatch配對建構器

本節說明配對建構器的關鍵元素，以及如何為您的遊戲建立和自訂。這包括設定配對組態和配對規則集。

建立配對建構器是FlexMatch藍圖中的第一個步驟：

- 路線圖：將配對新增至Amazon GameLift Servers託管解決方案
- 路線圖：使用 建立獨立的配對解決方案 FlexMatch

FlexMatch 配對建構器會執行建立遊戲配對的工作。它會管理收到的配對請求集區、處理和選取玩家以尋找最佳的可能玩家群組，並組成隊伍進行配對。對於使用 Amazon GameLift Servers 託管的遊戲，它也會放置並啟動配對的遊戲工作階段。

FlexMatch 會將配對服務與自訂規則引擎結合在一起。這樣您就可以設計如何依據對遊戲合理的玩家屬性及遊戲模式，將玩家配對在一起，並仰賴 FlexMatch 管理細節形成玩家群組，並將其置於遊戲之中。請參閱FlexMatch 規則集範例中的自訂配對詳細資訊。

形成配對後， FlexMatch會提供遊戲工作階段放置的配對資料。對於使用 Amazon GameLift Servers 託管的遊戲， FlexMatch 會將具有配對玩家的遊戲工作階段置放請求傳送至遊戲工作階段佇列。佇列會搜尋 Amazon GameLift Servers 機群可用的託管資源，並針對配對啟動新的遊戲工作階段。對於使用其他託管解決方案的遊戲， FlexMatch 會提供配對資料，供您提供給自己的遊戲工作階段置放元件。

如需 FlexMatch 配對建構器如何處理收到的配對請求詳細說明，請參閱 FlexMatch 配對程序。

主題

- 設計FlexMatch配對建構器
- 建置FlexMatch規則集
- 建立配對組態
- 設定FlexMatch事件通知

設計FlexMatch配對建構器

本主題提供如何設計適合您遊戲的配對建構器的指引。

主題

設定基本配對建構器

配對建構器至少需要下列元素：

- 規則集會為配對決定團隊的大小和範圍，並定義在為配對評估玩家時要使用的規則集。會將每個配對建構器設定為使用一個規則集。請參閱 建置FlexMatch規則集 和 FlexMatch 規則集範例。
- 通知目標會收到所有配對事件通知。您需要設定 Amazon Simple Notification Service (SNS) 主題，然後將主題 ID 新增至配對建構器。如需有關設定通知的詳細資訊，請參閱 設定FlexMatch事件通知。
- 請求逾時會決定配對建構請求在請求集區中保留的時間，並評估可能的配對。一旦請求已逾時，即無法進行配對並從集區中移除。
- FlexMatch 搭配Amazon GameLift Servers受管託管使用 時，遊戲工作階段佇列會尋找最佳可用資源來託管配對的遊戲工作階段，並啟動新的遊戲工作階段。每個佇列都會設定位置和資源類型清單（包括 Spot 或隨需執行個體），以決定遊戲工作階段可以放置的位置。如需佇列的詳細資訊，請參閱使用多位置佇列。

選擇配對建構器的位置

決定您希望配對活動在何處進行，並在該位置建立配對組態和規則集。 Amazon GameLift Servers會維護遊戲配對請求的票證集區，在其中進行排序和評估是否有可行的配對。進行配對後， 會Amazon GameLift Servers傳送遊戲工作階段放置的配對詳細資訊。您可以在託管解決方案支援的任何位置執行相符的遊戲工作階段。

如需可建立FlexMatch資源的位置，FlexMatch 支援 AWS 區域請參閱 。

AWS 區域 為您的配對建構器選擇 時，請考慮位置如何影響效能，以及它如何最佳化玩家的配對體驗。建議遵循下列最佳實務：

- 將配對建構器放置在靠近您的玩家和傳送FlexMatch配對請求的用戶端服務的位置。此方法可減少配對請求工作流程的延遲影響，並使其更有效率。

- 如果您的遊戲觸及全球觀眾，請考慮在多個位置建立配對建構器，並將配對請求路由到最接近玩家的配對建構器。除了提高效率之外，這還會導致票證集區與地理位置接近的玩家形成，這改善了配對建構器根據延遲要求配對玩家的能力。
- FlexMatch 搭配Amazon GameLift Servers受管託管使用 時，請將您的配對建構器及其使用的遊戲工作階段佇列放在相同的位置。這有助於將配對建構器和佇列之間的通訊延遲降到最低。

新增選用元素

除了這些最低要求以外，您可以使用下列額外選項來設定配對建構器。如果您使用 FlexMatch搭配 Amazon GameLift Servers託管解決方案，則會內建許多功能。如果您使用 FlexMatch做為獨立配對服務，建議您將這些功能建置到您的系統中。

玩家接受

您可以設定配對建構器，要求所有選擇配對的玩家都必須接受參與。如果您的系統需要接受，所有玩家都必須選擇接受或拒絕提議的配對。配對必須接收到來自提議配對中所有玩家的接受，才能完成。如果任何玩家拒絕或無法接受配對，則會捨棄提議的配對，並依照下列方式處理票證。票證中所有玩家都接受配對的票證會傳回配對集區以繼續處理。至少有一個玩家拒絕配對或無法回應的票證會進入失敗狀態，且不再處理。玩家接受需要時間限制；所有玩家都必須在時間限制內接受提議的配對，配對才能繼續。

回填模式

使用FlexMatch回填，讓您的遊戲工作階段在遊戲工作階段的整個生命週期內都充滿配對良好的新玩家。處理回填請求時， FlexMatch會使用與用來比對原始玩家的相同配對建構器。您可以自訂回填票證的優先順序，以及新配對的票證，將回填票證放在行的前面或結尾。這表示，當新玩家進入配對集區時，他們被放置在現有遊戲的可能性高於或低於新形成的遊戲。

無論您的遊戲FlexMatch搭配 受管Amazon GameLift Servers託管或其他託管解決方案使用 ，都可以手動回填。手動回填可讓您靈活地決定何時觸發回填請求。例如，您可能只想在遊戲的特定階段或特定條件存在時新增玩家。

自動回填僅適用於使用受管Amazon GameLift Servers託管的遊戲。啟用此功能後，如果遊戲工作階段從開啟的玩家位置開始， 會Amazon GameLift Servers開始自動為其產生回填請求。此功能可讓您設定配對，讓新遊戲以最少玩家數量開始，然後在新玩家進入配對集區時快速填滿。您可以在遊戲工作階段生命週期期間隨時關閉自動回填。

遊戲屬性

對於使用 FlexMatch搭配 Amazon GameLift Servers受管託管的遊戲，您可以提供其他資訊，以便在請求新的遊戲工作階段時傳遞至遊戲伺服器。這可能是傳遞遊戲模式組態的有用方法，這些組態是為要建立的配對類型啟動遊戲工作階段所需的。配對建構器建立的所有配對遊戲工作階段都會收到相同的一組遊戲屬性。您可以透過建立不同的配對組態來改變遊戲屬性資訊。

預留玩家空位

您可以指定要在每個配對中保留的特定玩家空位，並在稍後補上。您可以透過設定配對建構組態的「額外玩家數目」進行。

自訂事件資料

使用此屬性來為配對建構器包含在所有配對建構相關事件中的一組自訂資訊。在追蹤遊戲特定活動時 (包含追蹤配對建構器的效能)，此功能非常有用。

建置FlexMatch規則集

每個 FlexMatch 配對建構器都必須具有規則集。規則集判定配對的兩項關鍵要素：您的遊戲團隊結構及規模，以及如何將玩家分組在一起以達到最佳的配對。

例如規則集可能以下列方式說明配對：以兩個團隊建立配對，每個團隊各有五位玩家，其中一隊是防守者，另一隊則是入侵者。團隊可以有新手和經驗豐富的玩家，但兩個團隊的平均技能必須彼此相距 10 點以內。如果 30 秒後沒有找到配對，請逐漸放寬技巧要求。

本節主題說明如何設計和建置配對規則集。建立規則集時，您可以使用 Amazon GameLift Servers主控台或 AWS CLI。

主題

- 設計FlexMatch規則集
- 設計FlexMatch大型相符規則集
- 教學課程：建立配對規則集
- FlexMatch 規則集範例

設計FlexMatch規則集

本主題涵蓋規則集的基本結構，以及如何為最多 40 名玩家的小型配對建置規則集。配對規則集會執行兩件事：配置配對的團隊結構和大小，並告知配對建構器如何選擇玩家以形成最佳的可能配對。

但您的配對規則集可以執行更多操作。例如，您可以：

- 最佳化遊戲的配對演算法。
- 設定最低玩家延遲要求，以保護遊戲品質。
- 隨著時間逐漸放寬團隊要求和配對規則，讓所有作用中的玩家都能在需要時找到可接受的配對。
- 使用方彙總定義群組配對請求的處理。
- 處理 40 名或更多玩家的大型配對。如需建置大型配對的詳細資訊，請參閱 設計FlexMatch大型相符規則集。

建置配對規則集時，請考慮下列選用和必要的任務：

- 描述規則集（必要）
- 自訂比對演算法
- 宣告玩家屬性
- 定義配對團隊
- 設定玩家配對的規則
- 允許隨著時間的推移放寬需求

您可以使用 Amazon GameLift Servers主控台或 `CreateMatchmakingRuleSet`操作來建置規則集。

描述規則集（必要）

提供規則集的詳細資料。

- name（選用）– 供您自己使用的描述性標籤。此值與您使用 建立規則集時指定的規則集名稱無關 Amazon GameLift Servers。
- ruleLanguageVersion – 用來建立FlexMatch規則的屬性表達式語言版本。值必須為 `1.0`。

自訂比對演算法

FlexMatch 最佳化大多數遊戲的預設演算法，讓玩家以最短的等待時間進入可接受的配對。您可以自訂演算法並調整遊戲的配對。

以下是預設FlexMatch配對演算法：

1. FlexMatch 會將所有開啟的配對票證和回填票證放入票證集區。
2. FlexMatch 會將集區中的票證隨機分組為一或多個批次。隨著票證集區變大， 會FlexMatch形成額外的批次，以維持最佳的批次大小。

3. FlexMatch 會根據年齡在每個批次中排序票證。
4. FlexMatch 根據每個批次最舊的票證來建置相符項目。

若要自訂比對演算法，請將 algorithm元件新增至規則集結構描述。如需完整的參考資訊，FlexMatch 規則集結構描述請參閱 。

使用以下選用的自訂來影響配對程序的不同階段。

- 新增批次前排序
- 根據 batchDistance 屬性形成批次
- 排定回填票證的優先順序
- 偏好具有擴展的舊票證

新增批次前排序

您可以在形成批次之前排序票證集區。這種類型的自訂對具有大型票證集區的遊戲最有效。批次前排序有助於加速配對程序，並提高玩家在定義特性中的一致性。

使用演算法屬性 定義批次前排序方法batchingPreference。預設設定為 random。

自訂批次前排序的選項包括：

- 依玩家屬性排序。提供玩家屬性清單，以預先排序票證集區。

  若要依玩家屬性排序，請將 batchingPreference設定為 sorted，並在 中定義玩家屬性清單sortByAttributes。若要使用 屬性，請先在規則集的 playerAttributes元件中宣告 屬性。

  在下列範例中， 會根據玩家偏好的遊戲地圖來FlexMatch排序票證集區，然後依玩家技能來排序。產生的批次更有可能包含想要使用相同映射的類似熟練玩家。

  ```
  "algorithm": {
      "batchingPreference": "sorted",
      "sortByAttributes": ["map", "player_skill"],
      "strategy": "exhaustiveSearch"
  },
  ```

- 依延遲排序。建立具有最低可用延遲的相符項目，或快速建立具有可接受延遲的相符項目。此自訂適用於規則集形成超過 40 名玩家的大型配對。

將演算法屬性strategy設定為 balanced。平衡策略會限制規則陳述式的可用類型。如需詳細資訊，請參閱設計FlexMatch大型相符規則集。

FlexMatch 根據玩家報告的延遲資料，以下列其中一種方式排序票證：

- 最低延遲位置。票證集區會依玩家報告其最低延遲值的位置預先排序。FlexMatch然後， 會將相同位置中低延遲的票證批次處理，進而提供更佳的遊戲體驗。它也會減少每個批次中的票證數量，因此配對可能需要更長的時間。若要使用此自訂，請將 batchingPreference設定為 fastestRegion，如下列範例所示。

```
"algorithm": {
    "batchingPreference": "fastestRegion",
    "strategy": "balanced"
},
```

- 可接受的延遲會快速相符。票證集區會依玩家回報可接受延遲值的位置預先排序。這會形成包含更多票證的較少批次。隨著每個批次中有更多票證，尋找可接受的相符項目更快。若要使用此自訂，請將 屬性設定為 batchingPreference largestPopulation，如下列範例所示。

```
"algorithm": {
    "batchingPreference": "largestPopulation",
    "strategy": "balanced"
},
```

> **Note**
>
> 平衡策略的預設值為 largestPopulation。

排定回填票證的優先順序

如果您的遊戲實作自動回填或手動回填，您可以根據請求類型自訂FlexMatch處理配對票證的方式。請求類型可以是新的比對或回填請求。根據預設， 會將這兩種類型的請求FlexMatch視為相同。

回填優先順序會影響 FlexMatch如何處理批次處理票證。回填優先順序需要規則集才能使用詳盡的搜尋策略。

FlexMatch 不符合多個回填票證。

若要變更回填票證的優先順序，請設定 屬性 backfillPriority。

- 先比對回填票證。在建立新的相符項目之前，此選項會嘗試比對回填票證。這表示傳入玩家加入現有遊戲的機率較高。

  如果您的遊戲使用自動回填，最好使用此項目。自動回填通常用於具有短遊戲工作階段和高玩家周轉率的遊戲。自動回填有助於這些遊戲形成最少的可行配對，並使其開始，同時FlexMatch搜尋更多玩家來填滿開放的插槽。

  將 backfillPriority 設定為 high。

  ```
  "algorithm": {
      "backfillPriority": "high",
      "strategy": "exhaustiveSearch"
  },
  ```

- 最後比對回填票證。此選項會忽略回填票證，直到評估所有其他票證為止。這表示當傳入玩家無法將其配對至新遊戲時， 會將他們FlexMatch回填至現有遊戲。

  當您想要使用回填做為最後機會選項，讓玩家進入遊戲時，例如沒有足夠的玩家來組成新的配對時，此選項非常有用。

  將 backfillPriority 設定為 low。

  ```
  "algorithm": {
      "backfillPriority": "low",
      "strategy": "exhaustiveSearch"
  },
  ```

偏好具有擴展的舊票證

當配對難以完成時，擴展規則會放寬配對條件。當部分完成配對中的票證達到特定年齡時， 會Amazon GameLift Servers套用擴展規則。票證的建立時間戳記會決定何時Amazon GameLift Servers套用規則；根據預設， 會FlexMatch追蹤最近相符票證的時間戳記。

若要在 FlexMatch 套用擴展規則時變更 ，請設定 屬性expansionAgeSelection，如下所示：

- 根據最新的票證展開。此選項會根據新增至潛在配對的最新票證套用擴展規則。每次FlexMatch符合新的票證時，就會重設時鐘。使用此選項時，產生的配對品質通常較高，但需要更長的時間才能配對；如果配對請求花費太長的時間才能配對，則配對請求可能會在完成之前逾時。expansionAgeSelection 設定為 newest。newest預設為 。

- 根據最舊的票證展開。此選項會根據潛在配對中最舊的票證套用擴展規則。使用此選項時， 會更快速地FlexMatch套用擴展，從而改善最早配對玩家的等待時間，但會降低所有玩家的配對品質。將 `expansionAgeSelection` 設定為 `oldest`。

```
"algorithm": {
    "expansionAgeSelection": "oldest",
    "strategy": "exhaustiveSearch"
},
```

宣告玩家屬性

在本節中，列出要在配對請求中包含的個別玩家屬性。您可能會在規則集中宣告玩家屬性的兩個原因：

- 當規則集包含依賴玩家屬性的規則時。
- 當您想要透過配對請求將玩家屬性傳遞至遊戲工作階段時。例如，您可能想要在每個玩家連線之前，將玩家角色選擇傳遞至遊戲工作階段。

宣告玩家屬性時，請包括下列資訊：

- name （必要） – 此值對於規則集必須是唯一的。
- type （必要） – 屬性值的資料類型。有效資料類型為數字、字串、字串清單或字串映射。
- default （選用） – 如果配對請求不提供屬性值，請輸入要使用的預設值。如果未宣告預設值，且請求不包含值，則 FlexMatch 無法履行請求。

定義配對團隊

描述配對隊伍的結構和大小。每個配對都必須有至少一個隊伍，而且您可依需要定義任意數量的隊伍。所有隊伍可以有相同數量的玩家，也可以有數量不對等的玩家。例如，您可以定義一個單一玩家的怪物隊伍和一個有 10 名玩家的獵人隊伍。

FlexMatch 會根據規則集定義隊伍大小的方式，將配對請求處理為小型配對或大型配對。最多 40 名玩家的潛在配對是小型配對，超過 40 名玩家的配對是大型配對。若要確定規則集的潛在配對大小，請為規則集內定義的所有隊伍新增 maxPlayer 設定。

- name （必要） – 為每個團隊指派唯一的名稱。您可以在規則和擴展中使用此名稱，以及遊戲工作階段中配對資料的FlexMatch參考。
- maxPlayers （必要） – 指定要指派給團隊的玩家數量上限。

- minPlayers （必要） – 指定要指派給團隊的玩家人數下限。
- quantity （選用） – 指定使用此定義建立的團隊數量。當 FlexMatch建立配對時，它會為這些團隊提供具有附加號碼的名稱。例如 `Red-Team1`、 `Red-Team2`和 `Red-Team3`。

FlexMatch 會嘗試將隊伍填滿到玩家大小上限，但會建立較少玩家的隊伍。如果您希望配對內的所有隊伍有相同的大小，則可以對此建立規則。如需 `EqualTeamSizes`規則的範例，請參閱 FlexMatch 規則集範例主題。

設定玩家配對的規則

建立一組規則陳述式，評估玩家是否接受配對。規則可能會設定適用於個別玩家、隊伍或整個配對的要求。Amazon GameLift Servers 在處理配對請求時，一開始會先在有空玩家集區中尋找停留時間最久的玩家，並圍繞該玩家來建置配對。如需建立FlexMatch規則的詳細說明，請參閱 FlexMatch 規則類型。

- name （必要） – 有意義的名稱，可唯一識別規則集中的規則。規則名稱也會於事件記錄及指標之中參照，追蹤與此規則相關的活動。
- description （選用） – 使用此元素附加任意格式的文字描述。
- type （必要） – 類型元素可識別處理規則時要使用的操作。每個規則類型都需要一組額外的屬性。請至 FlexMatch 規則語言 參閱有效規則類型和屬性的清單。
- 規則類型屬性 （可能需要） – 視定義的規則類型而定，您可能需要設定特定規則屬性。請至 FlexMatch 規則語言 進一步了解屬性以及如何使用 FlexMatch 屬性運算式語言。

允許隨著時間的推移放寬需求

當 FlexMatch 找不到相符項目時，擴展可讓您放寬一段時間內的規則條件。此功能可確保 FlexMatch 在無法進行完美配對時提供最佳的 。透過擴展放寬規則，您可以逐步擴展可接受的玩家集區。

當不完整配對中最新票證的存留期符合擴展等待時間時，擴展就會開始。當 將新票證FlexMatch新增至配對時，擴展等待時鐘可能會重設。您可以在規則集的 `algorithm`區段中自訂擴展開始的方式。

以下是逐步增加配對所需最低技能水準的擴展範例。規則集使用名為 SkillDelta 的距離規則陳述式，要求配對中的所有玩家彼此在 5 個技能層級內。如果十五秒內沒有進行新的配對，則此擴展會尋找 10 的技能水準差異，十秒後則會尋找 20 的差異。

```
"expansions": [{
        "target": "rules[SkillDelta].maxDistance",
        "steps": [{
            "waitTimeSeconds": 15,
```

```
            "value": 10
        }, {
            "waitTimeSeconds": 25,
            "value": 20
        }]
    }]
```

啟用自動回填的配對建構器不會太快放寬玩家計數要求。新的遊戲工作階段需要花幾秒鐘的時間才能啟動，並開始自動回填作業。更好的方法是在自動回填傾向於為您的遊戲啟動後開始擴展。擴展時間會根據您的團隊組成而有所不同，因此測試可以為您的遊戲尋找最佳的擴展策略。

設計FlexMatch大型相符規則集

如果您的規則集建立允許 41 到 200 名玩家的相符項目，則需要對規則集組態進行一些調整。這些調整會最佳化配對演算法，使其可以建立可行的大型配對，同時讓玩家等待時間縮短。因此，大型配對規則集會使用針對常見配對優先順序最佳化的標準解決方案來取代耗時的自訂規則。

以下是如何判斷是否需要針對大型比對最佳化規則集的方法：

1. 對於規則集中定義的每個團隊，取得 maxPlayer 的值，
2. 新增所有 maxPlayer 值。如果總計超過 40，表示您有一個大型比對規則集。

若要針對大型比對最佳化您的規則集，請進行如下所述的調整。請參閱 中大型比對規則集的結構描述，大型比對的規則集結構描述以及 中的規則集範例範例：建立大型比對。

自訂大型比對的比對演算法

如果尚未存在演算法元件，請將演算法元件新增至規則集。設定下列屬性。

- `strategy`（必要）– 將 `strategy` 屬性設定為「平衡」。此設定會觸發 FlexMatch進行額外的配對後檢查，以根據 `balancedAttribute` 屬性中定義的指定玩家屬性來尋找最佳團隊平衡。平衡策略取代了自訂規則建置平均配對團隊的需求。
- `balancedAttribute`（必要）– 識別在配對中平衡隊伍時要使用的玩家屬性。此屬性必須具有數值資料類型（雙或整數）。例如，如果您選擇平衡玩家技能， FlexMatch 會嘗試指派玩家，讓所有隊伍擁有盡可能平均配對的彙總技能等級。平衡屬性必須在規則集的玩家屬性中宣告。
- `batchingPreference`（選用）– 選擇您希望為玩家設定最低延遲配對的重點程度。此設定會影響在建置相符項目之前，配對票證的排序方式。選項包括：
  - 最大人口。 FlexMatch 允許配對使用集區中具有可接受延遲值的所有票證，在至少一個共同位置。因此，潛在的票證集區往往很大，因此更容易更快地填滿相符項目。玩家可能會放置在具有可

接受但不一定最佳延遲的遊戲中。如果未設定 batchingPreference 屬性，則當 strategy 設定為「平衡」時，這是預設行為。

- 最快的位置。 會根據報告最低延遲值的位置，FlexMatch預先排序集區中的所有票證。因此，配對通常會與在同一位置報告低延遲的玩家形成。同時，每個配對的潛在票證集區較小，這可能會增加填滿配對所需的時間。此外，由於延遲的優先順序較高，因此配對中的玩家在平衡屬性方面可能會有更大的差異。

下列範例會將比對演算法設定為行為如下：(1) 預先排序票證集區，以依其具有可接受延遲值的位置分組票證；(2) 形成排序票證的批次以進行比對；(3) 建立與批次票證的比對，並平衡團隊以平衡平均玩家技能。

```
"algorithm": {
    "strategy": "balanced",
    "balancedAttribute": "player_skill",
    "batchingPreference": "largestPopulation"
},
```

宣告玩家屬性

請務必宣告在規則集演算法中用作平衡屬性的玩家屬性。配對請求中應包含每個玩家的此屬性。您可以為玩家屬性提供預設值，但在提供玩家特定值時，屬性平衡效果最好。

定義團隊

在定義隊伍大小和結構時，其程序與小型配對相同，但 FlexMatch 填滿隊伍的方式會不同。這會影響只有部分填滿時配對可能看起來的樣子。您可能需要調整團隊大小下限以回應。

FlexMatch 在將玩家指派給隊伍時，會使用以下規則。首先：尋找尚未達到其玩家要求下限的隊伍。其次：在這些隊伍中，尋找有最多空位的隊伍。

若配對定義了多個大小一樣的隊伍，系統會循序地將玩家新增到每個隊伍，直到隊伍填滿。因此，配對中的隊伍一定有近乎相等的玩家人數，即使配對不完整也一樣。目前無法強制讓大型配對中的隊伍都有一樣的大小。若配對的隊伍大小不對稱，則程序會更複雜一些。在此案例中，玩家一開始會指派給擁有最多開放位置的最大隊伍。隨著開放的空位數量在所有隊伍中變得更加平均，玩家會進入較小的隊伍。

例如，假設您有一個規則集，其中包含三個團隊。紅色和藍色團隊都設定為 maxPlayers=10，minPlayers=5。綠色團隊設定為 maxPlayers=3， minPlayers=2。以下是填充順序：

1. 沒有團隊達到 minPlayers。紅色和藍色隊伍有 10 個空位，綠色隊伍則有 3 個。前 10 名玩家會指派給紅色和藍色隊伍 (每個隊伍各 5 名)。兩個團隊現在都已達到 minPlayers。

2. 綠色團隊尚未達到 minPlayers。接下來的 2 名玩家會指派給綠色隊伍。綠色團隊現在已達到 minPlayers。
3. 在 的所有隊伍中minPlayers，現在會根據開放的空位數指派其他玩家。紅色和藍色團隊各有 5 個開放插槽，而綠色團隊則有 1 個。接下來的 8 名玩家（各 4 名）會指派給紅藍隊伍。所有團隊現在都有 1 個開放槽。
4. 其餘的 3 個玩家位置會（每個 1 個）指派給團隊，沒有特定順序。

設定大型比對的規則

大型配對的配對主要依賴平衡策略和延遲批次最佳化。因此大多數的自訂規則都無法使用。不過，您可以納入下列類型的規則：

- 設定玩家延遲硬性限制的規則。使用latency規則類型搭配 屬性 maxLatency。請參閱延遲規則參考。以下範例會將玩家延遲上限設定為 200 毫秒：

```
"rules": [{
        "name": "player-latency",
        "type": "latency",
        "maxLatency": 200
    }],
```

- 根據指定玩家屬性中的親近度批次玩家的規則。這與將平衡屬性定義為大型相符演算法的一部分不同，該演算法著重於建置平均相符的團隊。此規則會根據指定屬性值中的相似性批次處理配對票證，例如初學者或專家技能，這往往會導致與指定屬性上緊密一致的配對玩家。使用batchDistance規則類型，識別以數字為基礎的屬性，並指定要允許的最寬範圍。請參閱批次距離規則參考。以下是呼叫配對玩家彼此處於一個技能水準的範例：

```
"rules": [{
        "name": "batch-skill",
        "type": "batchDistance",
        "batchAttribute": "skill",
        "maxDistance": 1
```

放寬大型比對需求

和小型配對一樣，您也可以使用擴展，在無法達成有效配對時，隨時間放寬配對要求。對於大型配對，您可以選擇放寬延遲規則或隊伍玩家計數。

如果您對大型配對使用自動配對回填功能，請避免過於快速地放寬玩家人數。FlexMatch 只會在遊戲工作階段開始後，才開始產生回填請求，因此可能不會在建立配對後的幾秒鐘就發生。在這段時間內，FlexMatch 可能會建立多個部分填滿的新遊戲工作階段，尤其是當玩家人數規則調降時。因此，您最終會有比所需數量還多的遊戲工作階段，而讓玩家稀疏地分散到這些工作階段。最佳實務是讓玩家人數擴展的第一個步驟有較長的等待時間，且時間長到足以讓遊戲工作階段開始。由於大型配對的回填請求有較高的優先順序，因此在新的遊戲開始之前，系統會將進入的玩家安插到現有遊戲。您可能需要進行實驗以找出遊戲的理想等待時間。

以下範例會使用較長的初始等待時間來逐步降低黃色隊伍的玩家人數。請記住，規則集擴展的等待時間是絕對值，不能使用複合值。因此，第一個擴展發生在 5 秒時，第二個擴展則發生在 5 秒之後，也就是 10 秒時。

```
"expansions": [{
        "target": "teams[Yellow].minPlayers",
        "steps": [{
            "waitTimeSeconds": 5,
            "value": 8
        }, {
            "waitTimeSeconds": 10,
            "value": 5
        }]
    }]
```

教學課程：建立配對規則集

在您為配對Amazon GameLift ServersFlexMatch建構器建立配對規則集之前，建議您檢查規則集語法。使用Amazon GameLift Servers主控台或 AWS Command Line Interface (AWS CLI) 建立規則集後，您無法變更規則集。

請注意，您可以在 區域中擁有的規則集數目上限有服務配額 AWS ，因此最好刪除未使用的規則集。

Console

建立規則集

1. 在 https：//https://console.aws.amazon.com/gamelift/ 開啟 Amazon GameLift Servers主控台。
2. 切換到您要建立規則集 AWS 的區域。在與使用規則的配對組態相同的區域中定義規則集。
3. 在導覽窗格中，選擇 FlexMatch、配對規則集。
4. 在配對規則集頁面上，選擇建立規則集。

5. 在建立配對規則集頁面上，執行下列動作：

   a. 在規則集設定下，針對名稱輸入唯一的描述性名稱，可用於在清單或事件和指標資料表中識別它。

   b. 針對規則集，以 JSON 輸入您的規則集。如需設計規則集的資訊，請參閱 設計FlexMatch規則集。您也可以從 使用其中一個範例規則集FlexMatch 規則集範例。

   c. 選擇驗證以驗證規則集的語法是否正確。您無法在規則集建立後對其進行編輯，因此最好先進行驗證。

   d. （選用）在標籤下，新增標籤以協助您管理和追蹤 AWS 資源。

6. 選擇建立。如果建立成功，您可以搭配配對建構器使用規則集。

AWS CLI

建立規則集

開啟命令列視窗，並使用命令 create-matchmaking-rule-set。

此範例命令會建立簡單的配對規則集，以設定單一團隊。請務必在與使用它的配對組態相同的 AWS 區域中建立規則集。

```
aws gamelift create-matchmaking-rule-set \
    --name "SampleRuleSet123" \
    --rule-set-body '{"name": "aliens_vs_cowboys", "ruleLanguageVersion": "1.0",
 "teams": [{"name": "cowboys", "maxPlayers": 8, "minPlayers":  4}]}'
```

如果建立請求成功， 會Amazon GameLift Servers傳回包含您指定設定的 MatchmakingRuleSet 物件。配對建構器現在可以使用新的規則集。

Console

刪除規則集

1. 在 https：//https://console.aws.amazon.com/gamelift/ 開啟 Amazon GameLift Servers主控台。
2. 切換到您在其中建立規則集的區域。
3. 在導覽窗格中，選擇 FlexMatch、配對規則集。
4. 在配對規則集頁面上，選取您要刪除的規則集，然後選擇刪除。

5. 在刪除規則集對話方塊中，選擇刪除以確認刪除。

> **Note**
>
> 如果配對組態正在使用規則集， Amazon GameLift Servers會顯示錯誤訊息 (無法刪除規則集)。如果發生這種情況，請將配對組態變更為使用不同的規則集，然後再試一次。若要了解哪些配對組態正在使用規則集，請選擇規則集的名稱以檢視其詳細資訊頁面。

AWS CLI

刪除規則集

開啟命令列視窗，並使用命令 delete-matchmaking-rule-set 刪除配對規則集。

如果配對組態正在使用規則集， 會Amazon GameLift Servers傳回錯誤訊息。如果發生這種情況，請將配對組態變更為使用不同的規則集，然後再試一次。若要取得哪些配對組態正在使用規則集的清單，請使用 命令 describe-matchmaking-configurations 並指定規則集名稱。

此範例命令會檢查配對規則集的使用情況，然後刪除規則集。

```
aws gamelift describe-matchmaking-rule-sets \
    --rule-set-name "SampleRuleSet123" \
    --limit 10

aws gamelift delete-matchmaking-rule-set \
    --name  "SampleRuleSet123"
```

FlexMatch 規則集範例

FlexMatch 規則集可以涵蓋各種配對案例。下列的範例遵循 FlexMatch 組態架構和屬性表達式語言。請根據需要完整複製這些規則集，或選擇其中的元件。

關於使用 FlexMatch 規則與規則集，詳細資訊請參閱下列主題：

> **Note**
>
> 在評估包含多個玩家的配對單時，請求中的所有玩家皆必須符合配對要求。

主題

範例：建立兩個具有平均配對玩家的隊伍

此範例說明如何利用下列的指示，設定兩個勢均力敵的玩家隊伍。

- 建立兩個玩家隊伍。
  - 在每個隊伍中加入 4 到 8 個玩家。
  - 最終確定的隊伍必須擁有相同的玩家人數。
- 納入玩家的技能等級 (如果未提供，預設為 10)。
- 根據其技能等級是否類似於其他玩家的這項條件，來選擇玩家。確保兩個隊伍的玩家平均技能值落差在 10 點之內。
- 如果未能快速地完成配對，請放寬對玩家技能的要求，以在合理的時間內完成配對。
  - 經過 5 秒之後，請展開搜尋，以允許玩家平均技能值在 50 點之內的隊伍。
  - 經過 15 秒之後，請展開搜尋，以允許玩家平均技能值在 100 點之內的隊伍。

使用此規則集的注意事項：

- 此範例可讓隊伍的人數介於 4 到 8 名玩家之間 (雖然每個隊伍的人數必須相同)。對於人數在有效範圍內的隊伍，配對建構器會盡力嘗試，來配對最多的允許玩家人數。

- FairTeamSkill 規則可確保隊伍根據玩家的技能均衡配對。為了針對每個潛在的新玩家評估此項規則，FlexMatch 會暫時性地將玩家加入隊伍，並計算平均值。如果不符規則，則不會將潛在的玩家加入配對。
- 由於這兩個隊伍有相同結構，您可以選擇只建立一個隊伍定義，並將隊伍數量設定為「2」。在這個案例中，如果您將隊伍命名為「aliens」，則系統會對這兩個隊伍指派「aliens_1」和「aliens_2」名稱。

```
{
    "name": "aliens_vs_cowboys",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number",
        "default": 10
    }],
    "teams": [{
        "name": "cowboys",
        "maxPlayers": 8,
        "minPlayers": 4
    }, {
        "name": "aliens",
        "maxPlayers": 8,
        "minPlayers": 4
    }],
    "rules": [{
        "name": "FairTeamSkill",
        "description": "The average skill of players in each team is within 10 points
 from the average skill of all players in the match",
        "type": "distance",
        // get skill values for players in each team and average separately to produce
 list of two numbers
        "measurements": [ "avg(teams[*].players.attributes[skill])" ],
        // get skill values for players in each team, flatten into a single list, and
 average to produce an overall average
        "referenceValue": "avg(flatten(teams[*].players.attributes[skill]))",
        "maxDistance": 10 // minDistance would achieve the opposite result
    }, {
        "name": "EqualTeamSizes",
        "description": "Only launch a game when the number of players in each team
 matches, e.g. 4v4, 5v5, 6v6, 7v7, 8v8",
        "type": "comparison",
```

```
        "measurements": [ "count(teams[cowboys].players)" ],
        "referenceValue": "count(teams[aliens].players)",
        "operation": "=" // other operations: !=, <, <=, >, >=
    }],
    "expansions": [{
        "target": "rules[FairTeamSkill].maxDistance",
        "steps": [{
            "waitTimeSeconds": 5,
            "value": 50
        }, {
            "waitTimeSeconds": 15,
            "value": 100
        }]
    }]
}
```

範例：建立不均勻的團隊（獵人與怪物）

此範例說明一種遊戲模式，其中由一組玩家獵殺一隻怪物。玩家可選擇獵人或怪物的角色。獵人們會針對想要面對的怪物，指定怪物的最低技能等級。獵人隊伍的人數下限可隨時間放寬，以完成配對。此情境會產生下列指示：

- 建立一個包含 5 位獵人的隊伍。
- 建立一個單獨的隊伍，其中只包含 1 隻怪物。
- 加入下列的玩家屬性：
  - 玩家的技能等級 (如果未提供，預設為 10)。
  - 玩家偏好的怪物技能等級 (如果未提供，預設為 10)。
  - 玩家是否想要擔任怪物 (如果未提供此屬性，預設為 0 或 false)。
- 根據下列的條件，選擇一位玩家來擔任怪物：
  - 玩家必須請求怪物的角色。
  - 此玩家必須符合或超過已加入獵人隊伍的玩家所偏好的最高技能等級。
- 根據下列的條件，選擇獵人隊伍的玩家：
  - 申請擔任怪物角色的玩家，無法加入獵人隊伍。
  - 如果怪物角色已找到人選，玩家所要求的怪物技能等級必須低於建議怪物的技能。
- 如果未能快速完成配對，請放寬獵人隊伍的人數下限，如下所示：
  - 超過 30 秒後，可讓遊戲在獵人隊伍只包含 4 個玩家的情況下開始。
  - 超過 60 秒後，可讓遊戲在獵人隊伍只包含 3 個玩家的情況下開始。

使用此規則集的注意事項：

- 透過建立獵人和怪物兩個不同的隊伍，您就可以根據不同的一組條件來評估成員資格。

```
{
    "name": "players_vs_monster_5_vs_1",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number",
        "default": 10
    },{
        "name": "desiredSkillOfMonster",
        "type": "number",
        "default": 10
    },{
        "name": "wantsToBeMonster",
        "type": "number",
        "default": 0
    }],
    "teams": [{
        "name": "players",
        "maxPlayers": 5,
        "minPlayers": 5
    }, {
        "name": "monster",
        "maxPlayers": 1,
        "minPlayers": 1
    }],
    "rules": [{
        "name": "MonsterSelection",
        "description": "Only users that request playing as monster are assigned to the
 monster team",
        "type": "comparison",
        "measurements": ["teams[monster].players.attributes[wantsToBeMonster]"],
        "referenceValue": 1,
        "operation": "="
    },{
        "name": "PlayerSelection",
        "description": "Do not place people who want to be monsters in the players
 team",
        "type": "comparison",
        "measurements": ["teams[players].players.attributes[wantsToBeMonster]"],
```

```
        "referenceValue": 0,
        "operation": "="
    },{
        "name": "MonsterSkill",
        "description": "Monsters must meet the skill requested by all players",
        "type": "comparison",
        "measurements": ["avg(teams[monster].players.attributes[skill])"],
        "referenceValue":
 "max(teams[players].players.attributes[desiredSkillOfMonster])",
        "operation": ">="
    }],
    "expansions": [{
        "target": "teams[players].minPlayers",
        "steps": [{
            "waitTimeSeconds": 30,
            "value": 4
        },{
            "waitTimeSeconds": 60,
            "value": 3
        }]
    }]
}
```

範例：設定團隊層級需求和延遲限制

此範例說明如何設置玩家隊伍，並針對每個隊伍而非每個玩家套用一組規則。範例中使用單一定義來建立三個勢均力敵的隊伍。範例中也會建立所有玩家的最大延遲。延遲最大值可隨時間放寬，以完成配對。此範例會列出下列指示：

- 建立三個玩家隊伍。
  - 在每個隊伍中加入 3 到 5 個玩家。
  - 最終確定的隊伍，必須包含相同或幾乎相同的玩家人數 (落差在一個人以內)。
- 加入下列的玩家屬性：
  - 玩家的技能等級 (如果未提供，預設為 10)。
  - 玩家的遊戲角色 (如果未提供，預設為「農夫」)。
- 根據其技能等級是否接近配對中其他玩家的這項條件，來選擇玩家。
  - 確保每個隊伍的玩家平均技能值落差在 10 點之內。
- 根據下列的「medic (醫生)」角色人數來限制隊伍：
  - 整個配對最多可有 5 個醫生 (medic)。

- 僅限回報的延遲時間在 50 毫秒以內的配對玩家。
- 如果未能快速完成配對，請放寬玩家延遲的要求，如下所示：
  - 超過 10 秒後，允許玩家的延遲值最高可到 100 ms。
  - 超過 20 秒後，允許玩家的延遲值最高可到 150 ms。

使用此規則集的注意事項：

- 此規則集可確保隊伍根據玩家的技能均衡配對。為了評估 `FairTeamSkill` 規則，FlexMatch 會暫時性地在隊伍中新增候選玩家，並計算隊伍玩家的平均技能。然後，會與兩個隊伍中玩家的平均技能進行比較。如果不符規則，則不會將潛在的玩家加入配對。
- 隊伍和配對層級的要求 (medic 的總人數) 會透過集合規則滿足。此規則類型需要所有玩家的角色屬性清單，並針對數目上限進行比對。利用 `flatten` 來建立所有隊伍中所有玩家的清單。
- 根據延遲來進行評估時，請注意下列事項：
  - 延遲資料會在配對請求中提供，此請求是 Player (玩家) 物件的一部分。此資料並非玩家屬性，所以不需要列為屬性。若要取得準確的延遲測量，請使用 Amazon GameLift Servers的 UDP ping 信標。這些端點可讓您測量玩家裝置與每個潛在託管位置之間的實際 UDP 網路延遲，從而做出比使用 ICMP ping 更準確的置放決策。如需使用 UDP ping 信標測量延遲的詳細資訊，請參閱 UDP ping 信標。
  - 配對建構器會根據區域來評估延遲。配對建構器會略過延遲值高於上限值的任何區域。若要符合配對接受的資格，玩家必須至少擁有一個延遲值低於上限值的區域。
  - 如果配對請求略過一個或多個玩家的延遲資料，則該請求會遭到所有配對拒絕。

```
{
    "name": "three_team_game",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number",
        "default": 10
    },{
        "name": "character",
        "type": "string_list",
        "default": [ "peasant" ]
    }],
    "teams": [{
        "name": "trio",
        "minPlayers": 3,
```

```
        "maxPlayers": 5,
        "quantity": 3
    }],
    "rules": [{
        "name": "FairTeamSkill",
        "description": "The average skill of players in each team is within 10 points
from the average skill of players in the match",
        "type": "distance",
        // get players for each team, and average separately to produce list of 3
        "measurements": [ "avg(teams[*].players.attributes[skill])" ],
        // get players for each team, flatten into a single list, and average to
produce overall average
        "referenceValue": "avg(flatten(teams[*].players.attributes[skill]))",
        "maxDistance": 10 // minDistance would achieve the opposite result
    }, {
        "name": "CloseTeamSizes",
        "description": "Only launch a game when the team sizes are within 1 of each
other.  e.g. 3 v 3 v 4 is okay, but not 3 v 5 v 5",
        "type": "distance",
        "measurements": [ "max(count(teams[*].players))"],
        "referenceValue": "min(count(teams[*].players))",
        "maxDistance": 1
    }, {
        "name": "OverallMedicLimit",
        "description": "Don't allow more than 5 medics in the game",
        "type": "collection",
        // This is similar to above, but the flatten flattens everything into a single
        // list of characters in the game.
        "measurements": [ "flatten(teams[*].players.attributes[character])"],
        "operation": "contains",
        "referenceValue": "medic",
        "maxCount": 5
    }, {
        "name": "FastConnection",
        "description": "Prefer matches with fast player connections first",
        "type": "latency",
        "maxLatency": 50
    }],
    "expansions": [{
        "target": "rules[FastConnection].maxLatency",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 100
        }, {
```

```
            "waitTimeSeconds": 20,
            "value": 150
        }]
    }]
}
```

範例：使用明確排序來尋找最佳相符項目

此範例會設定兩個隊伍的簡單配對，這兩個隊伍各包含三個玩家。此範例說明如何利用明確排序的規則，來協助您盡快找到最佳的可能配對。這些規則會排序所有作用中的配對票證，以根據特定金鑰需求建立最佳相符項目。此範例會依照下列指示實作：

- 建立兩個玩家隊伍。
- 在每個隊伍中加入 3 個玩家。
- 加入下列的玩家屬性：
  - 體驗等級 (如果未提供，預設為 50)。
  - 偏好的遊戲模式 (可以列出多個值) (如果未提供，預設為「coop (合作)」和「deathmatch (死鬥)」)。
  - 偏好的遊戲地圖，包括地圖名稱和偏好的加權 (如果未提供，預設為 `"defaultMap"`，使用加權 100)。
- 設定預先排序：
  - 根據玩家對於和主錨玩家相同遊戲地圖的偏好，來將玩家排序。玩家可以擁有多個最愛的遊戲地圖，所以這個範例使用偏好設定值。
  - 根據玩家的經驗等級與主錨玩家的接近程度，來將玩家排序。利用此種排序法，所有隊伍中的所有玩家就能夠盡可能地擁有相近的經驗值。
- 所有隊伍的所有玩家必須至少選擇一個共同的遊戲模式。
- 所有隊伍的所有玩家必須至少選擇一個共同的遊戲地圖。

使用此規則集的注意事項：

- 遊戲地圖的排序，會使用比較 mapPreference 屬性值的絕對排序法。由於這是規則集內的第一個規則，所以會優先執行此排序。
- 體驗排序則會使用距離排序來比較候選玩家的技能等級與主錨玩家的技能。
- 排序會依其在規則集內的列出順序來執行。在此情境中，會先根據遊戲地圖的偏好設定，然後再根據經驗等級，來將玩家排序。

```
{
    "name": "multi_map_game",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "experience",
        "type": "number",
        "default": 50
    }, {
        "name": "gameMode",
        "type": "string_list",
        "default": [ "deathmatch", "coop" ]
    }, {
        "name": "mapPreference",
        "type": "string_number_map",
        "default": { "defaultMap": 100 }
    }, {
        "name": "acceptableMaps",
        "type": "string_list",
        "default": [ "defaultMap" ]
    }],
    "teams": [{
        "name": "red",
        "maxPlayers": 3,
        "minPlayers": 3
    }, {
        "name": "blue",
        "maxPlayers": 3,
        "minPlayers": 3
    }],
    "rules": [{
        // We placed this rule first since we want to prioritize players preferring the
 same map
       "name": "MapPreference",
       "description": "Favor grouping players that have the highest map preference
 aligned with the anchor's favorite",
       // This rule is just for sorting potential matches.  We sort by the absolute
 value of a field.
       "type": "absoluteSort",
       // Highest values go first
       "sortDirection": "descending",
       // Sort is based on the mapPreference attribute.
       "sortAttribute": "mapPreference",
```

```
        // We find the key in the anchor's mapPreference attribute that has the highest
 value.
        // That's the key that we use for all players when sorting.
        "mapKey": "maxValue"
    }, {
        // This rule is second because any tie-breakers should be ordered by similar
 experience values
        "name": "ExperienceAffinity",
        "description": "Favor players with similar experience",
        // This rule is just for sorting potential matches.  We sort by the distance
 from the anchor.
        "type": "distanceSort",
        // Lowest distance goes first
        "sortDirection": "ascending",
        "sortAttribute": "experience"
    }, {
        "name": "SharedMode",
        "description": "The players must have at least one game mode in common",
        "type": "collection",
        "operation": "intersection",
        "measurements": [ "flatten(teams[*].players.attributes[gameMode])"],
        "minCount": 1
    }, {
        "name": "MapOverlap",
        "description": "The players must have at least one map in common",
        "type": "collection",
        "operation": "intersection",
        "measurements": [ "flatten(teams[*].players.attributes[acceptableMaps])"],
        "minCount": 1
    }]
}
```

範例：尋找跨多個玩家屬性的交集

此範例說明如何使用集合規則來尋找兩個或多個玩家屬性中的交集。使用集合規則時，您可以分別針對單一屬性和多個屬性，使用 `intersection` 操作和 `reference_intersection_count` 操作。

為了說明這項方法，此範例會根據玩家的角色偏好，來評估配對中的玩家。此範例的遊戲是「自由戰」的模式，配對中的所有玩家都是對手。每個玩家都會被要求 (1) 選擇自己的角色，以及 (2) 選擇自己想要對抗的角色。我們需要規則，來確保配對中每個玩家所使用的角色，都在其他所有玩家的偏好對手清單中。

範例規則集利用下列的角色來說明配對：

- 隊伍結構：一個隊伍 5 個玩家
- 玩家屬性：
  - myCharacter：玩家所選擇的角色。
  - preferredOpponents：列出玩家想要對抗的角色。
- 配對規則：如果每個使用中的角色，出現在每個玩家偏好的對手清單中，即可接受可能的配對。

為了實行配對規則，此範例使用具有下列屬性值的集合規則：

- 操作 – 使用 `reference_intersection_count`操作來評估測量值中的每個字串清單如何與參考值中的字串清單相交。
- 測量 – 使用 `flatten` 屬性表達式來建立字串清單，每個字串清單都包含一個玩家的 myCharacter 屬性值。
- 參考值 – 使用 `set_intersection` 屬性表達式來建立所有 preferredOpponents 屬性值的字串清單，這些值對配對中的每個玩家都是常見的。
- 限制 – `minCount` 設定為 1，以確保每個玩家選擇的字元（測量中的字串清單）至少符合所有玩家通用的其中一個偏好對手。（參考值中的字串）。
- 擴展 – 如果配對未在 15 秒內填滿，請放寬最小交集要求。

此規則的流程如下：

1. 將玩家加入配對候選中。會重新計算參考值 (字串清單)，以納入與新玩家偏好對手清單的交集。會重新計算衡量值 (字串清單)，以將新玩家所選擇的角色新增為字串清單。
2. Amazon GameLift Servers 會確認衡量值 (玩家所選擇的角色) 中的每個字串清單，是否與參考值 (玩家偏好的對手) 中至少一個字串有交集。在此範例中，由於衡量值中的每個字串清單中只包含一個值，因此交集為 0 或 1。
3. 如果衡量值中的任何字串清單並未和參考值的字串清單有交集，則不符此規則，會從候選配對中將此一新玩家移除。
4. 如果未能在 15 秒內完成配對，請放寬對手的配對標準，以補滿配對中剩下的玩家空位。

```
{
    "name": "preferred_characters",
    "ruleLanguageVersion": "1.0",

    "playerAttributes": [{
```

```
        "name": "myCharacter",
        "type": "string_list"
    }, {
        "name": "preferredOpponents",
        "type": "string_list"
    }],

    "teams": [{
        "name": "red",
        "minPlayers": 5,
        "maxPlayers": 5
    }],

    "rules": [{
        "description": "Make sure that all players in the match are using a character
 that is on all other players' preferred opponents list.",
        "name": "OpponentMatch",
        "type": "collection",
        "operation": "reference_intersection_count",
        "measurements": ["flatten(teams[*].players.attributes[myCharacter])"],
        "referenceValue":
 "set_intersection(flatten(teams[*].players.attributes[preferredOpponents]))",
        "minCount":1
    }],
    "expansions": [{
        "target": "rules[OpponentMatch].minCount",
        "steps": [{
            "waitTimeSeconds": 15,
            "value": 0
        }]
    }]
}
```

範例：比較所有玩家的屬性

此範例說明如何在一組玩家之間比較玩家的屬性。

範例規則集利用下列的角色來說明配對：

- 隊伍結構：兩個單一玩家的隊伍
- 玩家屬性：
  - gameMode (遊戲模式)：玩家所選擇的遊戲類型 (如果未提供，預設為「回合制」)。

  - gameMap (遊戲地圖)：玩家所選擇的遊戲世界 (如果未提供，預設為 1)。
  - character (角色)：玩家所選擇的角色 (無預設值表示玩家必須指定角色)。
- 配對規則：配對的玩家必須符合下列要求：
  - 玩家必須選擇相同的遊戲模式。
  - 玩家必須選擇相同的遊戲地圖。
  - 玩家必須選擇不同的角色。

使用此規則集的注意事項：

- 為了實行配對規則，此範例使用比較規則來檢查所有玩家的屬性值。針對遊戲模式和地圖，該規則會驗證值是否相同。針對角色，該規則會驗證值是否不同。
- 此範例使用一個有數量屬性的玩家定義來建立這兩個玩家隊伍。隊伍會獲得以下名稱：「player_1」和「player_2」。

```
{
    "name": "",
    "ruleLanguageVersion": "1.0",

    "playerAttributes": [{
        "name": "gameMode",
        "type": "string",
        "default": "turn-based"
    }, {
        "name": "gameMap",
        "type": "number",
        "default": 1
    }, {
        "name": "character",
        "type": "number"
    }],

    "teams": [{
        "name": "player",
        "minPlayers": 1,
        "maxPlayers": 1,
        "quantity": 2
    }],
```

```
    "rules": [{
        "name": "SameGameMode",
        "description": "Only match players when they choose the same game type",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[gameMode])"]
    }, {
        "name": "SameGameMap",
        "description": "Only match players when they're in the same map",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[gameMap])"]
    }, {
        "name": "DifferentCharacter",
        "description": "Only match players when they're using different characters",
        "type": "comparison",
        "operation": "!=",
        "measurements": ["flatten(teams[*].players.attributes[character])"]
    }]
}
```

範例：建立大型比對

此範例說明如何為可以超過 40 名玩家的配對設定規則集。當規則集描述隊伍的 maxPlayer 計數總和大於 40 時，便會視為大型配對來處理。請至 設計FlexMatch大型相符規則集 進一步了解。

範例規則集使用以下指示來建立配對：

- 建立一個有玩家人數多達 200 名的隊伍，且玩家人數下限要求為 175 名。
- 平衡條件：根據類似技能等級來選取玩家。所有玩家都必須回報其技能等級才能進行配對。
- 批次處理偏好設定：在建立配對時，依照類似的平衡條件來將玩家群組在一起。
- 延遲規則：將可接受的玩家延遲上限設定為 150 毫秒。
- 如果未能快速填滿配對，則放寬要求以在合理的時間內完成配對。
  - 在 10 秒後，接受有 150 名玩家的隊伍。
  - 在 12 秒後，將可接受的延遲上限提高到 200 毫秒。
  - 在 15 秒後，接受有 100 名玩家的隊伍。

使用此規則集的注意事項：

- 由於演算法使用「最大人口」批次處理偏好設定，系統會先根據平衡條件來對玩家進行排序。因此，配對內往往會填滿玩家，並包含技能更類似的玩家。所有玩家皆符合可接受的延遲要求，但可能無法在所在位置獲得最佳的延遲。
- 這個規則集內所使用的演算法策略「最大人口」為預設設定。若要使用預設設定，您可以選擇省略該設定。
- 如果您已啟用配對回填，則請勿太快放寬玩家人數要求，否則最終可能會有太多部份填滿的遊戲工作階段。請至 放寬大型比對需求 進一步了解。

```
{
    "name": "free-for-all",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "skill",
        "type": "number"
    }],
    "algorithm": {
        "balancedAttribute": "skill",
        "strategy": "balanced",
        "batchingPreference": "largestPopulation"
    },
    "teams": [{
        "name": "Marauders",
        "maxPlayers": 200,
        "minPlayers": 175
    }],
    "rules": [{
        "name": "low-latency",
        "description": "Sets maximum acceptable latency",
        "type": "latency",
        "maxLatency": 150
    }],
    "expansions": [{
        "target": "rules[low-latency].maxLatency",
        "steps": [{
            "waitTimeSeconds": 12,
            "value": 200
        }],
    }, {
        "target": "teams[Marauders].minPlayers",
        "steps": [{
            "waitTimeSeconds": 10,
```

```
            "value": 150
        }, {
            "waitTimeSeconds": 15,
            "value": 100
        }]
    }]
}
```

範例：建立多團隊大型配對

此範例說明如何為有多個可以超過 40 名玩家的隊伍配對設定規則集。此範例說明如何使用一個定義建立多個相同隊伍，以及在建立配對期間會如何填滿大小不對稱的隊伍。

範例規則集使用以下指示來建立配對：

- 建立十個有多達 15 名玩家的相同「獵人」隊伍，以及一個有正好 5 名玩家的「怪物」隊伍。
- 平衡條件：根據擊殺怪物的數量來選取玩家。如果玩家未回報擊殺數量，則使用預設值 (5 個)。
- 批次處理偏好設定：根據回報的玩家延遲最快的區域來將玩家群組在一起。
- 延遲規則：將可接受的玩家延遲上限設定為 200 毫秒。
- 如果未能快速填滿配對，則放寬要求以在合理的時間內完成配對。
  - 在 15 秒後，接受有 10 名玩家的隊伍。
  - 在 20 秒後，接受有 8 名玩家的隊伍。

使用此規則集的注意事項：

- 此規則集定義了可以容納最多 155 名玩家的隊伍，這使其成為大型配對。(10 x 15 獵人 + 5 怪物 = 155)
- 由於演算法會使用「最快區域」批次處理偏好設定，系統可能會將玩家放到回報的延遲最快的區域，而非回報的延遲較高 (但可接受) 的區域。同時，配對可能會有較少的玩家，且平衡條件 (怪物技能數字) 可能會有更大的變化。
- 如果多隊伍定義 (數量 > 1) 已定義了擴展，擴展便會適用於使用該定義所建立的所有隊伍。因此，透過放寬獵人隊伍玩家人數下限設定，這十個獵人隊伍全部會受到同樣的影響。
- 由於此規則集已經過優化而能盡量減少玩家延遲，因此延遲規則可以一網打盡地排除連線選項未達接受標準的玩家。我們不需要放寬此要求。
- 以下是 FlexMatch 在擴展生效前針對此規則集填滿配對的方式：
  - 還沒有任何隊伍達到 minPlayers 人數。獵人隊伍有 15 個空位，怪物隊伍則有 5 個空位。

    - 前 100 名玩家會指派給十個獵人隊伍 (每個隊伍各 10 名)。
    - 接下來的 22 名玩家則會循序指派給獵人隊伍和怪物隊伍 (每個隊伍各 2 名)。
  - 獵人隊伍已達到 minPlayers 人數，也就是每個隊伍皆有 12 名玩家。怪物隊伍有 2 名玩家，且尚未達到 minPlayers 人數。
    - 接下來的三名玩家會指派給怪物隊伍。
  - 所有隊伍皆已達到 minPlayers 人數。獵人隊伍各有三個空位。怪物隊伍人數已滿。
    - 最後 30 名玩家會循序指派給獵人隊伍，以確保所有獵人隊伍的大小幾乎相同 (多一名玩家或少一名玩家)。
- 如果您已針對使用此規則集所建立的配對啟用回填，則請勿太快放寬玩家人數要求，否則最終可能會有太多部份填滿的遊戲工作階段。請至 放寬大型比對需求 進一步了解。

```
{
    "name": "monster-hunters",
    "ruleLanguageVersion": "1.0",
    "playerAttributes": [{
        "name": "monster-kills",
        "type": "number",
        "default": 5
    }],
    "algorithm": {
        "balancedAttribute": "monster-kills",
        "strategy": "balanced",
        "batchingPreference": "fastestRegion"
    },
    "teams": [{
        "name": "Monsters",
        "maxPlayers": 5,
        "minPlayers": 5
    }, {
        "name": "Hunters",
        "maxPlayers": 15,
        "minPlayers": 12,
        "quantity": 10
    }],
    "rules": [{
        "name": "latency-catchall",
        "description": "Sets maximum acceptable latency",
        "type": "latency",
        "maxLatency": 150
```

```
    }],
    "expansions": [{
        "target": "teams[Hunters].minPlayers",
        "steps": [{
            "waitTimeSeconds": 15,
            "value": 10
        }, {
            "waitTimeSeconds": 20,
            "value": 8
        }]
    }]
}
```

範例：與具有類似屬性的玩家建立大型配對

此範例說明如何使用 為兩個隊伍的配對設定規則集batchDistance。在範例中：

- 此SimilarLeague規則可確保配對中的所有玩家在其他玩家的 2 league 個內都有 。
- 此SimilarSkill規則可確保配對中的所有玩家在 10 個其他玩家skill中有 。如果玩家正在等待 10 秒，則距離會擴展到 20 秒。如果玩家正在等待 20 秒，則距離會擴展到 40 秒。
- 此SameMap規則可確保配對中的所有玩家都請求相同的 map。
- 此SameMode規則可確保配對中的所有玩家都請求相同的 mode。

```
{
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "name": "red",
        "minPlayers": 100,
        "maxPlayers": 100
    }, {
        "name": "blue",
        "minPlayers": 100,
        "maxPlayers": 100
    }],
    "algorithm": {
        "strategy":"balanced",
        "balancedAttribute": "skill",
        "batchingPreference":"fastestRegion"
    },
    "playerAttributes": [{
        "name": "league",
```

```
        "type": "number"
    },{
        "name": "skill",
        "type": "number"
    },{
        "name": "map",
        "type": "string"
    },{
        "name": "mode",
        "type": "string"
    }],
    "rules": [{
        "name": "SimilarLeague",
        "type": "batchDistance",
        "batchAttribute": "league",
        "maxDistance": 2
    }, {
        "name": "SimilarSkill",
        "type": "batchDistance",
        "batchAttribute": "skill",
        "maxDistance": 10
    }, {
        "name": "SameMap",
        "type": "batchDistance",
        "batchAttribute": "map"
    }, {
        "name": "SameMode",
        "type": "batchDistance",
        "batchAttribute": "mode"
    }],
    "expansions": [{
        "target": "rules[SimilarSkill].maxDistance",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 20
        }, {
            "waitTimeSeconds": 20,
            "value": 40
        }]
    }]
}
```

範例：使用複合規則來建立與具有類似屬性或類似選擇之玩家的配對

此範例說明如何使用 為兩個隊伍的配對設定規則集compound。在範例中：

- 此SimilarLeagueDistance規則可確保配對中的所有玩家在其他玩家的 2 league 個內都有 。
- 此SimilarSkillDistance規則可確保配對中的所有玩家在 10 個其他玩家skill中有 。如果玩家正在等待 10 秒，則距離會擴展到 20 秒。如果玩家正在等待 20 秒，則距離會擴展到 40 秒。
- 此SameMapComparison規則可確保配對中的所有玩家都請求相同的 map。
- 此SameModeComparison規則可確保配對中的所有玩家都請求相同的 mode。
- 如果至少符合下列其中一個條件，則CompoundRuleMatchmaker規則可確保相符：
  - 配對中的玩家已請求相同的 map和相同的 mode。
  - 配對中的玩家具有相當的 skill和 league 屬性。

```
{
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "name": "red",
        "minPlayers": 10,
        "maxPlayers": 20
    }, {
        "name": "blue",
        "minPlayers": 10,
        "maxPlayers": 20
    }],
    "algorithm": {
        "strategy":"balanced",
        "balancedAttribute": "skill",
        "batchingPreference":"fastestRegion"
    },
    "playerAttributes": [{
        "name": "league",
        "type": "number"
    },{
        "name": "skill",
        "type": "number"
    },{
        "name": "map",
        "type": "string"
    },{
        "name": "mode",
```

```
        "type": "string"
    }],
    "rules": [{
        "name": "SimilarLeagueDistance",
        "type": "distance",
        "measurements": ["max(flatten(teams[*].players.attributes[league]))"],
        "referenceValue": "min(flatten(teams[*].players.attributes[league]))",
        "maxDistance": 2
    }, {
        "name": "SimilarSkillDistance",
        "type": "distance",
        "measurements": ["max(flatten(teams[*].players.attributes[skill]))"],
        "referenceValue": "min(flatten(teams[*].players.attributes[skill]))",
        "maxDistance": 10
    }, {
        "name": "SameMapComparison",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[map])"]
    }, {
        "name": "SameModeComparison",
        "type": "comparison",
        "operation": "=",
        "measurements": ["flatten(teams[*].players.attributes[mode])"]
    }, {
        "name": "CompoundRuleMatchmaker",
        "type": "compound",
        "statement": "or(and(SameMapComparison, SameModeComparison),
 and(SimilarSkillDistance, SimilarLeagueDistance))"
    }],
    "expansions": [{
        "target": "rules[SimilarSkillDistance].maxDistance",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 20
        }, {
            "waitTimeSeconds": 20,
            "value": 40
        }]
    }]
}
```

範例：建立使用玩家封鎖清單的規則

此範例說明規則集，可讓玩家避免與其他特定玩家配對。玩家可以建立封鎖清單，配對建構器會在玩家選擇配對期間評估該清單。如需新增封鎖清單或避免清單功能的詳細資訊，請參閱AWS 遊戲部落格的。

此範例會列出下列指示：

- 建立兩個隊伍，只有五個玩家。
- 傳入玩家的封鎖清單，這是玩家 IDs的清單 （最多 100 個）。
- 將所有玩家與每個玩家的封鎖清單進行比較，如果找到任何封鎖的玩家 IDs，則會拒絕提議的配對。

使用此規則集的注意事項：

- 評估新玩家以新增至提議配對 （或回填現有配對中的位置） 時，玩家可能會因為下列其中一個原因遭到拒絕：
  - 如果新玩家位於已為配對選取的任何玩家的封鎖清單上。
  - 如果任何已為配對選取的玩家在新玩家的封鎖清單中。
- 如圖所示，此規則集可防止將玩家與區塊清單上的任何玩家配對。您可以新增規則擴展並增加`maxCount`值，將此需求變更為偏好設定 （也稱為「避免」清單）。

```
{
    "name": "Player Block List",
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "red"
    }, {
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "blue"
    }],
    "playerAttributes": [{
        "name": "BlockList",
        "type": "string_list",
        "default": []
    }],
    "rules": [{
```

```
        "name": "PlayerIdNotInBlockList",
        "type": "collection",
        "operation": "reference_intersection_count",
        "measurements": "flatten(teams[*].players.attributes[BlockList])",
        "referenceValue": "flatten(teams[*].players[playerId])",
        "maxCount": 0
    }]
}
```

建立配對組態

若要設定Amazon GameLift ServersFlexMatch配對建構器來處理配對請求，請建立配對組態。使用 Amazon GameLift Servers主控台或 AWS Command Line Interface (AWS CLI)。如需建立配對建構器的詳細資訊，請參閱 設計FlexMatch配對建構器。

主題

- 教學課程：建立用於Amazon GameLift Servers託管的配對建構器
- 教學課程：為獨立 建立配對建構器 FlexMatch
- 教學課程：編輯配對組態

教學課程：建立用於Amazon GameLift Servers託管的配對建構器

在建立配對組態之前，請建立規則集和Amazon GameLift Servers遊戲工作階段佇列，以便與配對建構器搭配使用。

Console

1. 在Amazon GameLift Servers主控台的導覽窗格中，選擇配對組態。
2. 切換到您要建立配對建構器 AWS 的區域。
3. 在配對組態頁面上，選擇建立配對組態。
4. 在定義組態詳細資訊頁面的配對組態詳細資訊下，執行下列動作：

    a. 在名稱中，輸入配對建構器名稱，協助您在清單和指標中識別它。配對建構器名稱在 區域中必須是唯一的。配對請求會依名稱和區域識別要使用的配對建構器。

    b. （選用）針對描述，新增描述以協助識別配對建構器。

    c. 針對規則集，從清單中選擇要與配對建構器搭配使用的規則集。清單包含您在目前區域中建立的所有規則集。

d. 針對 FlexMatch 模式，選擇受管以進行Amazon GameLift Servers受管託管。此模式FlexMatch會提示將成功配對傳遞至指定的遊戲工作階段佇列。

e. 針對 AWS 區域，選擇您設定要與配對建構器搭配使用之遊戲工作階段佇列的區域。

f. 針對佇列，選擇您要與配對建構器搭配使用的遊戲工作階段佇列。

5. 選擇下一步。

6. 在設定設定頁面的配對設定下，執行下列動作：

a. 針對請求逾時，設定配對建構器完成每個請求配對的時間上限，以秒為單位。 會FlexMatch取消超過此時間的配對請求。

b. 針對回填模式，選擇處理配對回填的模式。

- 若要開啟自動回填功能，請選擇自動。
- 若要建立您自己的回填請求管理或不使用回填功能，請選擇手動。

c. （選用）對於其他玩家計數，設定在配對中保持開啟的玩家位置數量。 FlexMatch可以在未來向玩家填滿這些位置。

d. （選用）在配對接受選項下，對於需要接受，如果您想要要求提議配對中的每個玩家主動接受配對，請選取需要。如果您選取此選項，則對於接受逾時，請設定配對建構器在取消配對之前等待玩家接受的時間，以秒為單位。

7. （選用）在事件通知設定下，執行下列動作：

a. （選用）針對 SNS 主題，選擇 Amazon Simple Notification Service (Amazon SNS) 主題以接收配對事件通知。如果您尚未設定 SNS 主題，稍後可以透過編輯配對組態來選擇此選項。如需詳細資訊，請參閱設定FlexMatch事件通知。

b. （選用）對於自訂事件資料，請在事件傳訊中輸入任何您想要與此配對建構器相關聯的自訂資料。 會在與配對建構器相關聯的每個事件中FlexMatch包含此資料。

8. （選用）展開其他遊戲資料，然後執行下列動作：

a. （選用）對於遊戲工作階段資料，輸入FlexMatch您希望交付給新遊戲工作階段的任何其他遊戲相關資訊，這些工作階段是從使用此配對組態建立的配對開始。

b. （選用）對於遊戲屬性，新增包含新遊戲工作階段相關資訊的鍵/值對屬性。

9. （選用）在標籤下，新增標籤以協助您管理和追蹤 AWS 資源。

10. 選擇下一步。

11. 在檢閱和建立頁面上，檢閱您的選擇，然後選擇建立。成功建立後，配對建構器已準備好接受配對請求。

AWS CLI

若要使用 建立配對組態 AWS CLI，請開啟命令列視窗，並使用 create-matchmaking-configuration 命令來定義新的配對建構器。

此範例命令會建立新的配對組態，需要玩家接受並啟用自動回填。它還保留兩個玩家位置 FlexMatch供 稍後新增玩家，並提供一些遊戲工作階段資料。

```
aws gamelift create-matchmaking-configuration \
    --name "SampleMatchamker123" \
    --description "The sample test matchmaker with acceptance" \
    --flex-match-mode WITH_QUEUE \
    --game-session-queue-arns "arn:aws:gamelift:us-
west-2:111122223333:gamesessionqueue/MyGameSessionQueue" \
    --rule-set-name "MyRuleSet" \
    --request-timeout-seconds 120 \
    --acceptance-required \
    --acceptance-timeout-seconds 30 \
    --backfill-mode AUTOMATIC \
    --notification-target "arn:aws:sns:us-
west-2:111122223333:My_Matchmaking_SNS_Topic" \
    --additional-player-count 2 \
    --game-session-data "key=map,value=winter444"
```

若配對組態建立請求成功，Amazon GameLift Servers 即會傳回 MatchmakingConfiguration 物件，其涵蓋您所請求的配對建構器設定。新的配對建構器已準備好接受配對請求。

教學課程：為獨立 建立配對建構器 FlexMatch

建立配對組態之前，請建立規則集以搭配配對建構器使用。

Console

1. 在 https：//https://console.aws.amazon.com/gamelift/home 開啟 Amazon GameLift Servers主控台。
2. 切換到您要建立配對建構器 AWS 的區域。如需支援FlexMatch配對組態的區域清單，請參閱選擇配對建構器的位置。
3. 在導覽窗格中，選擇 FlexMatch、 Matchmaking 組態。
4. 在配對組態頁面上，選擇建立配對組態。
5. 在定義組態詳細資訊頁面的配對組態詳細資訊下，執行下列動作：

a. 在名稱中，輸入配對建構器名稱，協助您在清單和指標中識別它。配對建構器名稱在 區域中必須是唯一的。配對請求會依名稱和區域識別要使用的配對建構器。

b. （選用）針對描述，新增描述以協助識別配對建構器。

c. 針對規則集，從清單中選擇要與配對建構器搭配使用的規則集。清單包含您在目前區域中建立的所有規則集。

d. 針對 FlexMatch 模式，選擇獨立。這表示您有自訂機制，可在 外部的託管解決方案上啟動新的遊戲工作階段Amazon GameLift Servers。

6. 選擇下一步。

7. 在設定設定頁面的配對設定下，執行下列動作：

a. 針對請求逾時，設定配對建構器完成每個請求配對的時間上限，以秒為單位。超過此時間的配對請求會遭到拒絕。

b. （選用）在配對接受選項下，對於需要接受，如果您想要要求提議配對中的每個玩家主動接受配對，請選取需要。如果您選取此選項，則對於接受逾時，請設定配對建構器在取消配對之前等待玩家接受的時間，以秒為單位。

8. （選用）在事件通知設定下，執行下列動作：

a. （選用）針對 SNS 主題，選擇 Amazon SNS 主題以接收配對事件通知。如果您尚未設定 SNS 主題，稍後可以透過編輯配對組態來選擇此選項。如需詳細資訊，請參閱設定FlexMatch事件通知。

b. （選用）對於自訂事件資料，請在事件傳訊中輸入任何您想要與此配對建構器相關聯的自訂資料。 會在與配對建構器相關聯的每個事件中FlexMatch包含此資料。

9. （選用）在標籤下，新增標籤以協助您管理和追蹤 AWS 資源。

10. 選擇下一步。

11. 在檢閱和建立頁面上，檢閱您的選擇，然後選擇建立。成功建立後，配對建構器已準備好接受配對請求。

AWS CLI

若要使用 建立配對組態 AWS CLI，請開啟命令列視窗，並使用 create-matchmaking-configuration 命令來定義新的配對建構器。

此範例命令會為需要玩家接受的獨立配對建構器建立新的配對組態。

```
aws gamelift create-matchmaking-configuration \
```

```
    --name "SampleMatchamker123" \
    --description "The sample test matchmaker with acceptance" \
    --flex-match-mode STANDALONE \
    --rule-set-name "MyRuleSetOne" \
    --request-timeout-seconds 120 \
    --acceptance-required \
    --acceptance-timeout-seconds 30 \
    --notification-target "arn:aws:sns:us-
west-2:111122223333:My_Matchmaking_SNS_Topic"
```

若配對組態建立請求成功，Amazon GameLift Servers 即會傳回 MatchmakingConfiguration 物件，其涵蓋您所請求的配對建構器設定。新的配對建構器已準備好接受配對請求。

教學課程：編輯配對組態

若要編輯配對組態，請從導覽列中選擇配對組態，然後選擇您要編輯的組態。您可以更新現有組態中的任何欄位，但其名稱除外。

更新組態規則集時，如果有現有的作用中配對票證，新的規則集可能會不相容，原因如下：

- 新的或不同的團隊名稱或團隊數量
- 新的玩家屬性
- 現有玩家屬性類型的變更

若要對規則集進行任何這些變更，請使用更新的規則集建立新的配對組態。

設定FlexMatch事件通知

您可以使用事件通知來追蹤個別配對請求的狀態。生產中或生產前具有大量配對活動的所有遊戲都應使用事件通知。

設定事件通知有兩種選項。

- 讓您的配對建構器將事件通知發佈至 Amazon Simple Notification Service (Amazon SNS) 主題。
- 使用自動發佈的 Amazon EventBridge 事件及其工具套件來管理事件。

如需 Amazon GameLift Servers 發出的FlexMatch事件清單，請參閱 FlexMatch 配對事件。

> **Important**
>
> 對於大量配對系統，我們建議使用標準（非 FIFO) Amazon SNS 主題，而不是 FIFO 主題。FIFO 主題的發佈限制低於標準主題，這可能會導致在高負載期間調節例外狀況。如果您使用 FIFO 主題遇到限流，您可能會遺失FlexMatch通知。

> **Note**
>
> Amazon GameLift Servers 會使用內建的重試邏輯自動處理 Amazon SNS 交付失敗和限流。當 Amazon SNS 傳回限流錯誤或暫時失敗時， Amazon GameLift Servers 會在嘗試之間以漸進式延遲重試通知交付。這有助於確保可靠地傳送事件通知。不過，如果在所有重試嘗試後仍持續失敗，或是授權失敗或遺失主題等無法重試的錯誤，則通知可能會遺失。

主題

設定 EventBridge 事件

Amazon GameLift Servers 會自動將所有配對事件發佈至 Amazon EventBridge。使用 EventBridge，您可以設定規則，將配對事件路由到目標進行處理。例如，您可以設定規則，將事件 "PotentialMatchCreated" 路由到處理玩家接受度的 AWS Lambda 函數。如需詳細資訊，請參閱什麼是 Amazon EventBridge？

> **Note**
>
> 當您設定配對建構器時，請保持通知目標欄位空白，或者如果您想要同時使用 EventBridge 和 Amazon SNS 主題。

教學課程：設定 Amazon SNS 主題

您可以讓 Amazon GameLift Servers 將FlexMatch配對建構器產生的所有事件發佈至 Amazon SNS 主題。

建立Amazon GameLift Servers事件通知的 SNS 主題

1. 開啟 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Topics (主題)。
3. 在 Topics (主題) 頁面上，選擇 Create topic (建立主題)。
4. 在 主控台中建立主題。如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的使用 建立主題 AWS 管理主控台。
5. 在主題的詳細資訊頁面上，選擇編輯。
6. （選用）在主題的編輯頁面上，展開存取政策，然後從下列 AWS Identity and Access Management (IAM) 政策陳述式將粗體語法新增至現有政策的結尾。（為了清楚起見，此處會顯示整個政策。）請務必將 Amazon Resource Name (ARN) 詳細資訊用於您自己的 SNS 主題和 Amazon GameLift Servers配對組態。

   JSON

```
{
  "Version":"2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
```

```
      ],
      "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
      "Condition": {
        "StringEquals": {
        "AWS:SourceAccount": "111122223333"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-1:111122223333:your_topic_name",
      "Condition": {
        "ArnLike": {
        "aws:SourceArn": "arn:aws:gamelift:us-
east-1:111122223333:matchmakingconfiguration/your_configuration_name"
        }
      }
    }
  ]
}
```

7. 選擇儲存變更。

使用伺服器端加密設定 SNS 主題

您可以使用伺服器端加密 (SSE) 將敏感資料存放在加密主題中。SSE 使用在 AWS Key Management Service () 中管理的金鑰來保護 Amazon SNS 主題中的訊息內容AWS KMS。如需使用 Amazon SNS 進行伺服器端加密的詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的靜態加密。

若要使用伺服器端加密設定 SNS 主題，請檢閱下列主題：

- 《 AWS Key Management Service 開發人員指南》中的建立金鑰
- 《Amazon Simple Notification Service 開發人員指南》中的為主題啟用 SSE

建立 KMS 金鑰時，請使用下列 KMS 金鑰政策：

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "gamelift.amazonaws.com"
  },
  "Action": [
      "kms:Decrypt",
      "kms:GenerateDataKey"
  ],
  "Resource": "*",
  "Condition": {
      "ArnLike": {
        "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:matchmakingconfiguration/
your_configuration_name"
      },
      "StringEquals": {
        "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:your_account:your_sns_topic_name"
      }
  }
}
```

設定主題訂閱以叫用 Lambda 函數

您可以使用發佈至 Amazon SNS 主題的事件通知來叫用 Lambda 函數。設定配對建構器時，請務必將通知目標設定為 SNS 主題的 ARN。

下列 AWS CloudFormation 範本會將名為 的 SNS 主題訂閱設定為`MyFlexMatchEventTopic`叫用名為 的 Lambda 函數`FlexMatchEventHandlerLambdaFunction`。範本會建立 IAM 許可政策，Amazon GameLift Servers允許 寫入 SNS 主題。範本接著會新增 SNS 主題的許可，以叫用 Lambda 函數。

```
FlexMatchEventTopic:
  Type: "AWS::SNS::Topic"
  Properties:
    KmsMasterKeyId: alias/aws/sns #Enables server-side encryption on the topic using an
 AWS managed key
    Subscription:
      - Endpoint: !GetAtt FlexMatchEventHandlerLambdaFunction.Arn
        Protocol: lambda
    TopicName: MyFlexMatchEventTopic
```

```
FlexMatchEventTopicPolicy:
  Type: "AWS::SNS::TopicPolicy"
  DependsOn: FlexMatchEventTopic
  Properties:
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: Allow
          Principal:
            Service: gamelift.amazonaws.com
          Action:
            - "sns:Publish"
          Resource: !Ref FlexMatchEventTopic
    Topics:
      - Ref: FlexMatchEventTopic

FlexMatchEventHandlerLambdaPermission:
  Type: "AWS::Lambda::Permission"
  Properties:
    Action: "lambda:InvokeFunction"
    FunctionName: !Ref FlexMatchEventHandlerLambdaFunction
    Principal: sns.amazonaws.com
    SourceArn: !Ref FlexMatchEventTopic
```

為 準備遊戲 FlexMatch

使用 Amazon GameLift Servers FlexMatch 將玩家配對功能新增至您的遊戲。您可以FlexMatch搭配受管Amazon GameLift Servers託管解決方案使用 ，或搭配其他託管解決方案做為獨立服務使用。如果您想要將 FlexMatch新增至Amazon GameLift ServersFleetIQ解決方案，請使用它做為獨立服務。如何 FlexMatch 運作方式的詳細資訊，請參閱 如何Amazon GameLift Servers FlexMatch運作。

配對解決方案需要下列工作：

- 使用自訂配對規則建立配對建構器 如需建立配對建構器的詳細資訊，請參閱綁定Amazon GameLift Servers FlexMatch配對建構器。
- 更新您的遊戲用戶端，以允許玩家請求配對。
- 對於使用Amazon GameLift Servers託管的遊戲，請更新您的遊戲伺服器以管理配對資料，並選擇性地在配對上回填空的插槽。

本節中的主題涵蓋如何將配對支援新增至遊戲用戶端和遊戲伺服器。

請參閱您偏好的FlexMatch配對解決方案藍圖：

- 路線圖：將配對新增至Amazon GameLift Servers託管解決方案
- 路線圖：使用 建立獨立的配對解決方案 FlexMatch

FlexMatch 新增至遊戲用戶端

本主題說明如何將FlexMatch配對功能新增至用戶端遊戲元件。

我們強烈建議您的遊戲用戶端透過後端遊戲服務提出配對請求。透過使用此受信任來源與 Amazon GameLift Servers服務進行通訊，您可以更輕鬆地防止駭客嘗試和仿造玩家資料。如果您的遊戲有工作階段目錄服務，這是處理配對請求的理想選擇。FlexMatch 搭配 Amazon GameLift Servers託管和獨立服務使用 時，使用後端遊戲服務來呼叫 Amazon GameLift Servers 服務是最佳實務。

無論您是將 FlexMatch與Amazon GameLift Servers受管託管搭配使用，還是將 作為獨立服務與另一個託管解決方案搭配使用，都需要用戶端更新。使用 Amazon GameLift ServersSDK 一部分的 服務 API AWS ，新增下列功能：

- 為一或多個玩家請求配對 （必要）。根據您的配對規則集，此請求可能需要特定玩家特定的資料，包括玩家屬性和延遲。

- 追蹤配對請求的狀態（必要）。一般而言，此任務需要設定事件通知。
- 請求玩家接受提議的配對（選用）。此功能需要額外與玩家的互動，才能顯示配對詳細資訊，並允許他們接受或拒絕配對。
- 取得遊戲工作階段連線資訊並加入遊戲（必要）。為新配對啟動遊戲工作階段後，請擷取遊戲工作階段的連線資訊，並使用它來連線至遊戲工作階段。

先決條件用戶端任務

您必須先執行下列任務，才能將用戶端功能新增至遊戲：

- 將 AWS SDK 新增至您的後端服務。您的後端服務使用 Amazon GameLift Servers API 中的功能，這是 AWS SDK 的一部分。請參閱Amazon GameLift Servers用戶端服務的 SDKs，以進一步了解 AWS SDK 並下載最新版本。如需 API 說明和功能，請參閱 Amazon GameLift Servers FlexMatch API 參考 (AWS SDK)。
- 設定配對票證系統。所有配對請求都必須有唯一的票證 ID。建立產生唯一票證 IDs機制，並將其指派給符合請求。票證 ID 可使用任何字串格式，最長可達 128 個字元。
- 收集配對建構器的相關資訊。從配對組態和規則集取得以下資訊。
  - 配對組態資源的名稱。
  - 玩家屬性清單，在規則集中定義。
- 擷取玩家資料。設定取得每個玩家相關資料的方法，以包含在配對請求中。您需要玩家 ID 和玩家屬性值。如果您的規則集有延遲規則，或您想要在放置遊戲工作階段時使用延遲資料，請收集玩家可能進入遊戲的每個地理位置的延遲資料。若要取得準確的延遲測量，請使用 Amazon GameLift Servers 的 UDP ping 信標。這些端點可讓您測量玩家裝置與每個潛在託管位置之間的實際 UDP 網路延遲，從而做出比使用 ICMP ping 更準確的置放決策。如需使用 UDP ping 信標測量延遲的詳細資訊，請參閱 UDP ping 信標。

為玩家請求配對

將程式碼新增至遊戲後端服務，以管理配對FlexMatch建構器的配對請求。對於FlexMatch搭配 Amazon GameLift Servers 託管使用的遊戲，以及FlexMatch做為獨立解決方案使用的遊戲，請求FlexMatch配對的程序相同。

若要建立配對請求：

呼叫 Amazon GameLift Servers API StartMatchmaking. 每個請求都必須包含下列資訊。

配對建構器

要在請求時使用的配對組態名稱。FlexMatch 會將每個請求置放在特定配對建構器的集區中，並根據配對建構器配置方式來處理請求。這包括強制執行時間限制，無論是否請求玩家接受配對，放置產生的遊戲工作階段時要使用哪個佇列等。進一步了解規則建置器和規則集：設計FlexMatch配對建構器。

票證 ID

指派給請求的唯一票證 ID。與請求相關的所有內容 (包含事件和通知) 將會參考票證 ID。

玩家資料

您想要為其建立配對的玩家清單。根據配對規則與延遲最低限制，如果請求中有任何玩家不符合配對規定，配對請求就絕不會加以配對。配對請求中最多可加入 10 名玩家。如果請求中有多名玩家，FlexMatch 會嘗試建立單一配對，並將所有玩家指派給同一個隊伍 (隨機選取)。如果請求包含玩家人數過多而無法全部加入其中一個配對隊伍，則請求將無法配對。舉例來說，如果您已將配對建構器設定為建立二對二配對 (兩個隊伍，每隊兩名玩家)，則您無法傳送包含超過兩名玩家的配對請求。

> **Note**
>
> 玩家 (由玩家 ID 加以識別) 一次僅能加入一個進行中的配對請求。當您為玩家建立了新請求，任何含有相同玩家 ID 的啟用中配對票證均會自動取消。

對於每個列出的玩家，請包含下列資料：

- 玩家 ID – 每個玩家都必須擁有您產生的唯一玩家 ID。請參閱產生玩家 IDs。

  > **Important**
  >
  > 當您建立新的配對請求，其中包含已包含在現有作用中配對請求中的玩家 ID 時，現有的請求會自動取消。不過，不會針對已取消的請求傳送`MatchmakingCancelled`事件。若要監控現有配對請求的狀態，請使用 DescribeMatchmaking 以不常間隔輪詢請求狀態 (30-60 秒)。取消的請求會顯示 狀態`CANCELLED`，其中包含原因 `Cancelled due to duplicate player`。

- 玩家屬性 – 如果 中的配對建構器使用對玩家屬性的呼叫，則請求必須為每個玩家提供這些屬性。配對規則集會定義必要的玩家屬性，其中也會指定屬性的資料類型。只有當規則集指定屬性的預

設值時，玩家屬性才會是選用的。如果配對請求未提供所有玩家的必要玩家屬性，則配對請求永遠無法成功。在建置FlexMatch規則集和FlexMatch 規則集範例中進一步了解配對建置規則集和玩家屬性。

- 玩家延遲 – 如果使用的配對建構器具有玩家延遲規則，則請求必須報告每個玩家的延遲。玩家延遲資料是每個玩家的一或多個值的清單。其代表玩家在配對建構器佇列中區域的玩家體驗延遲。如果請求中沒有包含玩家的延遲值，則無法對玩家進行配對，請求也會失敗。若要取得準確的延遲測量，請使用 Amazon GameLift Servers的 UDP ping 信標。這些端點可讓您測量玩家裝置與潛在託管位置之間的實際 UDP 網路延遲，從而做出比使用 ICMP ping 更準確的置放決策。如需使用 UDP ping 信標測量延遲的詳細資訊，請參閱 UDP ping 信標。

擷取相符請求詳細資訊

傳送配對請求後，您可以使用請求的票證 ID 呼叫 DescribeMatchmaking 來檢視請求詳細資訊。此呼叫會傳回請求資訊，包含目前的狀態。成功完成請求之後，票證也會包含遊戲用戶端連線至配對所需的資訊。

若要取消比對請求

您只要呼叫 StopMatchmaking，即可隨時取消配對請求。

追蹤配對事件

設定通知以追蹤 Amazon GameLift Servers 針對配對程序發出的事件。您可以直接設定通知、建立 SNS 主題或使用 Amazon EventBridge。如需有關設定通知的詳細資訊，請參閱 設定FlexMatch事件通知。設定好通知後，在用戶端服務上新增接聽程式，以偵測事件並依需求回應。

當一大段時間過去而完全不通知的情況下，定期輪詢狀態更新來備份通知，也是個不錯的主意。若要盡量降低對於配對效能的影響，請務必在配對單提交後至少 30 秒或最後收到通知後才進行輪詢。

使用請求的票證 ID 呼叫 DescribeMatchmaking，擷取配對請求票證，其中包含目前的狀態。建議輪詢不要超過 10 秒一次。這個方法僅適用於低用量開發案例。

> **Note**
>
> 在進行大量配對使用之前 (例如進行生產前負載測試)，您應該使用事件通知來設定遊戲。公開發行版本中的所有遊戲都應該使用通知，不論用量如何。持續輪詢方式只適用於開發時使用少量配對的遊戲。

請求玩家接受

如果您正在使用玩家已開啟接受的遊戲建置器，則將程式碼新增到您的用戶端服務，以管理玩家接受程序。對於FlexMatch搭配 Amazon GameLift Servers受管託管使用的遊戲，以及FlexMatch做為獨立解決方案使用的遊戲，管理玩家接受的程序相同。

請求玩家接受建議配對：

1. 當提議配對需要玩家接受時進行偵測。當狀態變更為 REQUIRES_ACCEPTANCE 時，監控配對票證以進行偵測。此狀態的變更會觸發FlexMatch事件 MatchmakingRequiresAcceptance。
2. 獲得所有玩家接受。建立機制，以便向每個玩家出示配對票證中的提議配對詳細資訊。玩家必須能夠表示他們接受或拒絕提議的配對。您可以呼叫 DescribeMatchmaking 來擷取配對詳細資訊。玩家須在有限時間內回應，否則配對建置器會撤銷提議的配對並繼續下一步。
3. 向 FlexMatch 回報玩家回應。呼叫 AcceptMatch 以回報玩家的回覆設定，藉此了解該玩家接受或拒絕配對。配對請求中的所有玩家均需接受配對才能繼續。
4. 處理接受失敗的票證。當任何玩家在提議的配對中拒絕配對，或無法於接受時間限制內回覆，請求都會失敗。接受配對之玩家的門票會自動傳回至門票集區。未接受配對的玩家票證會移至失敗狀態，且不會再處理。對於有多個玩家的門票，如果門票中的任何玩家不接受配對，則整個門票會失敗。

連線至相符項目

將程式碼新增至您的用戶端服務，以處理成功形成的配對 （ 狀態COMPLETED或事件 MatchmakingSucceeded)。這包括通知配對的玩家並將連線資訊交付到其遊戲用戶端。

對於使用Amazon GameLift Servers受管託管的遊戲，當成功滿足配對請求時，遊戲工作階段連線資訊會新增至配對票證。呼叫 DescribeMatchmaking，擷取完整的配對票證。連線資訊包含遊戲工作階段的 IP 地址和連接埠，以及每個玩家 ID 的玩家工作階段 ID。請至 GameSessionConnectionInfo 進一步了解。您的遊戲用戶端可以使用此資訊直接連線至遊戲工作階段以進行配對。連線請求應包含玩家工作階段 ID 和玩家 ID。此資料會將連線的玩家與遊戲工作階段的配對資料產生關聯，其中包含團隊指派 (請參閱 GameSession)。

對於使用其他託管解決方案的遊戲，包括Amazon GameLift Servers FleetIQ，您必須建置一種機制，讓配對玩家連線到適當的遊戲工作階段。

範例配對請求

下列程式碼片段會為數個不同的配對建構器建置配對請求。如上所述，請求必須提供使用中配對建置器所需的玩家屬性，如同配對建置器規則集所定義。提供的屬性必須使用相同的資料類型，如規則集中定義的數字 (N) 或字串 (S)。

```
# Uses matchmaker for two-team game mode based on player skill level
def start_matchmaking_for_cowboys_vs_aliens(config_name, ticket_id, player_id, skill,
 team):
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "skill": {"N": skill}
            },
            "PlayerId": player_id,
            "Team": team
        }],
        TicketId=ticket_id)

# Uses matchmaker for monster hunter game mode based on player skill level
def start_matchmaking_for_players_vs_monster(config_name, ticket_id, player_id, skill,
 is_monster):
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "skill": {"N": skill},
                "desiredSkillOfMonster": {"N": skill},
                "wantsToBeMonster": {"N": int(is_monster)}
            },
            "PlayerId": player_id
        }],
        TicketId=ticket_id)

# Uses matchmaker for brawler game mode with latency
def start_matchmaking_for_three_team_brawler(config_name, ticket_id, player_id, skill,
 role):
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "skill": {"N": skill},
```

```
                "character": {"S": [role]},
            },
            "PlayerId": player_id,
            "LatencyInMs": { "us-west-2": 20}
        }],
        TicketId=ticket_id)

# Uses matchmaker for multiple game modes and maps based on player experience
def start_matchmaking_for_multi_map(config_name, ticket_id, player_id, skill, maps,
 modes):
    response = gamelift.start_matchmaking(
        ConfigurationName=config_name,
        Players=[{
            "PlayerAttributes": {
                "experience": {"N": skill},
                "gameMode": {"SL": modes},
                "mapPreference": {"SL": maps}
            },
            "PlayerId": player_id
        }],
        TicketId=ticket_id)
```

FlexMatch 新增至 Amazon GameLift Servers託管的遊戲伺服器

當 Amazon GameLift Servers建立配對時，會產生一組配對結果資料，描述關鍵配對的詳細資訊，包括團隊指派。遊戲伺服器會在啟動新的遊戲工作階段來託管配對時，使用此資料和其他遊戲工作階段資訊。

對於使用 託管的遊戲伺服器 Amazon GameLift Servers

會Amazon GameLift Servers提示遊戲伺服器程序啟動遊戲工作階段。它提供 GameSession 物件，描述要建立的遊戲工作階段類型，並包含玩家特定資訊，包括配對資料。

對於在其他解決方案上託管的遊戲伺服器

成功完成配對請求後， Amazon GameLift Servers 會發出包含配對結果的事件。您可以將此資料與您自己的託管解決方案搭配使用，以啟動配對的遊戲工作階段。

關於配對建構器資料

比對資料包含下列資訊：

- 唯一的比對 ID
- 用來建立相符項目的配對設定 ID
- 為配對選取的玩家
- 團隊名稱和團隊指派
- 用來形成配對的玩家屬性值。屬性也可能提供指示遊戲工作階段設定方式的資訊。例如，遊戲伺服器可能會根據玩家屬性將角色指派給玩家，或選擇所有玩家通用的遊戲地圖偏好設定。或者，您的遊戲可能會根據平均玩家技能等級解鎖特定功能或等級。

配對資料不包含玩家延遲。如果您需要目前玩家的延遲資料，例如配對回填，我們建議您取得新的資料。

> **Note**
>
> 配對建構器資料會指定完整的配對組態 ARN，以識別組態名稱、 AWS 帳戶和區域。對於使用 託管的遊戲Amazon GameLift Servers，如果您使用配對回填，則只需要組態名稱。組態名稱是遵循 "：matchmakingconfiguration/" 的字串。在下列範例中，配對組態名稱為 "MyMatchmakerConfig"。

此 JSON 範例顯示典型的配對建構器資料集。它描述了一個兩玩家遊戲，玩家根據技能評分進行配對，並達到最高水準。

```
{
 "matchId":"1111aaaa-22bb-33cc-44dd-5555eeee66ff",
 "matchmakingConfigurationArn":"arn:aws:gamelift:us-
west-2:111122223333:matchmakingconfiguration/MyMatchmakerConfig",
 "teams":[
    {"name":"attacker",
  "players":[
           {"playerId":"4444dddd-55ee-66ff-77aa-8888bbbb99cc",
   "attributes":{
    "skills":{
     "attributeType":"STRING_DOUBLE_MAP",
     "valueAttribute":{"Body":10.0,"Mind":12.0,"Heart":15.0,"Soul":33.0}}
   }
  }]
 },{
  "name":"defender",
  "players":[{
```

```
    "playerId":"3333cccc-44dd-55ee-66ff-7777aaaa88bb",
    "attributes":{
     "skills":{
      "attributeType":"STRING_DOUBLE_MAP",
      "valueAttribute":{"Body":11.0,"Mind":12.0,"Heart":11.0,"Soul":40.0}}
    }
   }]
  }]
}
```

設定 的遊戲伺服器 FlexMatch

與 託管的遊戲伺服器Amazon GameLift Servers必須與Amazon GameLift Servers伺服器 SDK 整合，並具有核心功能，如新增至Amazon GameLift Servers遊戲伺服器中所述。此功能可讓您的遊戲伺服器在Amazon GameLift Servers託管資源上執行，並與 Amazon GameLift Servers服務通訊。下列指示說明新增FlexMatch功能所需的其他任務。

將 FlexMatch新增至您的遊戲伺服器

1. 啟動遊戲工作階段時使用配對資料。您的遊戲伺服器會實作名為 的回呼函數`onStartGameSession()`。建立配對後， Amazon GameLift Servers會尋找可用的遊戲伺服器程序，並呼叫此函數來提示它啟動配對的遊戲工作階段。此呼叫包含遊戲工作階段物件 (GameSession)。您的遊戲伺服器會使用遊戲工作階段資訊來啟動遊戲工作階段，包括配對建構器資料。如需啟動遊戲工作階段的詳細資訊，請參閱啟動遊戲工作階段。如需配對建構器資料的詳細資訊，請參閱關於配對建構器資料。
2. 處理玩家連線。連線到相符的遊戲時，遊戲用戶端會參考玩家 ID 和玩家工作階段 ID （請參閱驗證新玩家)。將遊戲伺服器設定為使用玩家 ID，將傳入玩家與配對建構器資料中的玩家資訊建立關聯。配對建構器資料可識別玩家的團隊指派和其他資訊，以代表遊戲中的玩家。
3. 玩家離開遊戲時進行回報。請確定您的遊戲伺服器呼叫伺服器 SDK RemovePlayerSession 來報告捨棄的玩家。如果您使用FlexMatch回填來填充現有遊戲中的空槽，此步驟特別重要。進一步了解如何在 中實作FlexMatch回填使用 回填現有遊戲 FlexMatch。
4. 請求新玩家填入現有的配對 （選用）。決定您要如何回填即時配對。如果您的配對建構器將回填模式設定為「手動」，您可能想要將回填支援新增至遊戲。如果回填模式設定為「自動」，您可能需要針對個別遊戲工作階段將其關閉。例如，在遊戲工作階段達到遊戲中的特定點後，您可能想要停止回填。進一步了解如何在 中實作配對回填使用 回填現有遊戲 FlexMatch。

使用 回填現有遊戲 FlexMatch

配對回填會使用 FlexMatch 機制，針對現有已配對的遊戲工作階段尋找新玩家。雖然您可以隨時將玩家新增至任何遊戲 （ 請參閱加入玩家到遊戲工作階段 ），但配對回填可確保新玩家符合與目前玩家相同的配對條件。此外，配對回填會將新玩家指定到團隊、管理玩家接受和將更新配對資訊傳送到遊戲伺服器。進一步了解配對回填：FlexMatch 配對程序。

> **Note**
>
> FlexMatch 回填目前不適用於使用 的遊戲Amazon GameLift Servers Realtime。

回填機制有兩種類型：

- 啟用自動回填以填滿開頭小於允許玩家上限的遊戲工作階段。自動回填不會回填加入遊戲然後退出的玩家。
- 設定手動回填機制，以取代正在退出遊戲工作階段的玩家。此機制必須能夠偵測開啟的插槽，並產生回填請求來填滿它。

開啟自動回填

使用自動配對回填功能，只要遊戲工作階段以一或多個未填滿的玩家位置開始，Amazon GameLift Servers 便會自動觸發回填請求。此功能可讓遊戲在找到最低匹配玩家人數後立即開始，並在配對到其他玩家時填滿剩餘的位置。您可以隨時選擇停止自動回填。

例如，假設遊戲可以容納 6 到 10 名玩家。 FlexMatch一開始會找到 6 名玩家，組成配對，並啟動新的遊戲工作階段。使用自動回填功能，新的遊戲工作階段可以立即請求額外的四名玩家。根據遊戲風格，我們可能想要允許新玩家在遊戲工作階段期間隨時加入。或者，我們可能想要在初始設定階段和遊戲開始之前停止自動回填。

若要將自動回填新增到您的遊戲，請對遊戲進行下列更新。

1. 啟用自動回填。自動回填是由配對組態加以管理。啟用時，自動回填會與該配對建置器建立的所有配對遊戲工作階段搭配使用。只要遊戲工作階段在遊戲伺服器上啟動，Amazon GameLift Servers 便會開始針對非完整遊戲工作階段產生回填請求。

   若要開啟自動回填，開放配對組態並將回填模式為「自動」(AUTOMATIC)。如需詳細資訊，請參閱 建立配對組態。

2. 開啟回填優先順序。自訂您的配對程序，在建立新的配對之前，先排定填補回填請求的優先順序。在您的配對規則集中，新增演算法元件並將回填優先順序設定為「高」。如需詳細資訊，請參閱自訂比對演算法。
3. 使用新的配對建構器資料更新遊戲工作階段。 會使用伺服器開發套件回呼函數，以配對資訊 Amazon GameLift Servers更新您的遊戲伺服器 onUpdateGameSession (請參閱初始化伺服器程序)。將程式碼新增到您的遊戲伺服器，以便在回填活動後處理更新的遊戲工作階段物件。請至更新遊戲伺服器上的配對資料 進一步了解。
4. 關閉遊戲工作階段的自動回填。在個別遊戲工作階段期間，您可以隨時停止自動回填。若要停止自動回填，將程式碼新增到您的遊戲用戶端或遊戲伺服器，讓 Amazon GameLift Servers API 呼叫 StopMatchmaking。此呼叫需要票證 ID。從最新的回填請求使用回填票證 ID。您可以從遊戲工作階段配對資料取得此資訊，更新內容如之前步驟所述。

從遊戲伺服器產生手動回填請求

您可以從託管遊戲工作階段的遊戲伺服器程序手動啟動配對回填請求。伺服器程序具有連接至遊戲之玩家up-to-date，以及空玩家位置的狀態。

本主題假設您已成功建置必要的 FlexMatch 元件並成功將配對建構程序新增至遊戲伺服器和用戶端遊戲服務。如需設定 FlexMatch 的詳細資訊，請參閱 路線圖：將配對新增至Amazon GameLift Servers託管解決方案。

若要為遊戲啟用配對回填，新增以下功能：

- 請將配對建構回填請求傳送到配對建構器來追蹤請求的狀態。
- 更新遊戲工作階段的配對資訊。請參閱 更新遊戲伺服器上的配對資料。

如同其他伺服器功能，遊戲伺服器會使用 Amazon GameLift Servers 伺服器開發套件。您可以 C++ 及 C # 使用此開發套件。

為了從遊戲伺服器進行配對回填，請完成以下任務。

1. 觸發配對回填請求。一般而言，當配對的遊戲有一或多個玩家空位時，建議您初始化回填請求。建議您將回填請求連繫至特定的情況 (例如填入關鍵字元角色或平衡團隊)。您可能還想要根據遊戲工作階段的存留期動限制回填活動。
2. 建立回填請求。您能夠新增程式碼以建立配對回填請求，並將其傳送至 FlexMatch 配對建構器。使用這些伺服器 API 處理回填請求：

- StartMatchBackfill()
- StopMatchBackfill()

若要建立回填請求，請以下列資訊呼叫 StartMatchBackfill。若要取消回填請求，請使用回填請求票證 ID 來呼叫 StopMatchBackfill。

- 票證 ID — 提供配對票證 ID （或選擇讓它們自動產生）。您可以使用相同的機制，將票證 ID 指派到配對建構和回填請求。會以相同的方式處理配對建構和回填的票證。
- 配對建構器 — 識別要用於回填請求的配對建構器。一般而言，您將想要使用用於建立原始配對的相同配對建構器。此請求需要配對建構組態 ARN。這項資訊會存放在遊戲工作階段物件 (GameSession) 中，啟用遊戲工作階段時，Amazon GameLift Servers 即會將該物件提供給伺服器程序。配對建構組態 ARN 包含在 MatchmakerData 屬性中。
- 遊戲工作階段 ARN — 識別要回填的遊戲工作階段。您可以呼叫伺服器 API GetGameSessionId() 來取得遊戲工作階段 ARN。在配對建構程序期間，新請求的票證不會有遊戲工作階段 ID，而回填請求的票證會有。在遊戲工作階段 ID 的存在可讓您了解新配對的票證和回填票證的差異。
- 玩家資料 — 在您回填的遊戲工作階段中包含所有目前玩家的玩家資訊 (Player)。此資訊可讓配對建構器為目前在遊戲工作階段中的玩家找到最可能的玩家配對。您必須包含每個玩家的團隊成員資格。如果您未使用回填，請勿指定團隊。如果您的遊戲伺服器已準確報告玩家連線狀態，您應該要能取得此資料，如下所示：

  1. 代管遊戲工作階段的伺服器程序應擁有哪些玩家正連接到遊戲工作階段的最新資訊。
  2. 若要取得玩家 IDs、屬性和團隊指派，請從遊戲工作階段物件 (GameSession)、MatchmakerData屬性 （請參閱）提取玩家資料關於配對建構器資料。配對建構器資料包含配對至遊戲工作階段的所有玩家，您只需要為目前連線的玩家提取玩家資料。
  3. 對於玩家延遲，如果配對建構器呼叫延遲資料，從所有目前玩家收集新延遲值，並將其包含在每個 Player 物件。如果將延遲資料省略和配對建構器有延遲規則，則請求無法成功相配。回填請求只需要遊戲工作階段目前所在之區域的延遲資料。您可以從 GameSession 物件的 GameSessionId 屬性取得遊戲工作階段區域，此值是包含在該區域的 ARN。

3. 追蹤回填請求的狀態。 會使用伺服器開發套件回呼函數Amazon GameLift Servers，更新您的遊戲伺服器關於回填請求的狀態 onUpdateGameSession（請參閱初始化伺服器程序)。在 新增程式碼以處理狀態訊息，以及由於成功回填請求而更新的遊戲工作階段物件更新遊戲伺服器上的配對資料。

配對建構器可隨時處理來自遊戲工作階段的配對回填請求。如果您需要取消請求，請呼叫 StopMatchBackfill()。如果您需要變更請求，請呼叫 StopMatchBackfill 接著提交更新請求。

從後端服務產生手動回填請求

除了從遊戲伺服器傳送回填請求，您可能要從用戶端遊戲服務來傳送他們。若要使用此選項，用戶端服務必須能夠存取遊戲工作階段活動和玩家連線上的目前資料；如果您的遊戲使用工作階段目錄服務，這可能會是最佳選擇。

本主題假設您已成功建置必要的 FlexMatch 元件並成功將配對建構程序新增至遊戲伺服器和用戶端遊戲服務。如需設定 FlexMatch 的詳細資訊，請參閱 路線圖：將配對新增至Amazon GameLift Servers 託管解決方案。

若要為遊戲啟用配對回填，新增以下功能：

- 請將配對建構回填請求傳送到配對建構器來追蹤請求的狀態。
- 更新遊戲工作階段的配對資訊。請參閱 更新遊戲伺服器上的配對資料

如同其他用戶端功能，用戶端遊戲服務會使用 AWS SDK 搭配 Amazon GameLift Servers API。SDK 有 C++、C# 及多種其他語言的選擇。如需用戶端 APIs 的一般說明，請參閱 Amazon GameLift Servers API 參考，其中說明 Amazon GameLift Servers動作的服務 API 和語言特定參考指南的連結。

若要設定用戶端遊戲服務，以回填配對遊戲，請完成以下任務。

1. 觸發回填請求。一般而言，當配對的遊戲有一或多個玩家空位時，遊戲會初始化回填請求。建議您將回填請求連繫至特定的情況 (例如填入關鍵字元角色或平衡團隊)。您可能還想要根據遊戲工作階段的存留期動限制回填。無論您用於觸發的條件為何，至少您會需要下列資訊。您可以透過呼叫含遊戲工作階段 ID 的 DescribeGameSessions 來從遊戲工作階段物件 (GameSession) 取得此資訊。

   - 目前玩家空位數。您可以從遊戲工作階段玩家上限和目前玩家數目來計算此值。每當遊戲伺服器聯絡 Amazon GameLift Servers 服務來驗證新玩家連線或報告捨棄的玩家時，即會更新目前玩家數目。

   - 建立政策。此設定表示遊戲工作階段目前是否接受新玩家。

遊戲工作階段物件包含其他可能有用的資訊 (包括遊戲工作階段開始時間、自訂遊戲屬性和配對建構器資料)。

2. 建立回填請求。您能夠新增程式碼以建立配對回填請求，並將其傳送至 FlexMatch 配對建構器。使用這些用戶端 API 處理回填請求：

   - StartMatchBackfill
   - StopMatchmaking

   若要建立回填請求，請以下列資訊呼叫 `StartMatchBackfill`。回填請求類似於配對建構請求 (請參閱 為玩家請求配對)，但其能識別現有的遊戲工作階段。若要取消回填請求，請使用回填請求票證 ID 來呼叫 `StopMatchmaking`。

   - 票證 ID — 提供配對票證 ID （或選擇讓它們自動產生）。您可以使用相同的機制，將票證 ID 指派到配對建構和回填請求。會以相同的方式處理配對建構和回填的票證。
   - 配對建構器 — 識別要使用的配對組態名稱。一般而言，您將想要使用用於建立原始配對之回填的相同配對建構器。此資訊會在配對建構組態 ARN 下的遊戲工作階段物件 (GameSession)、`MatchmakerData` 屬性中。名稱值是接在「matchmakingconfiguration/」之後的字串。(例如，在 ARN 值「arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MM-4v4」，配對建構組態名稱為「MM-4v4」。)
   - 遊戲工作階段 ARN — 指定要回填的遊戲工作階段。從遊戲工作階段物件使用 `GameSessionId` 屬性，此 ID 會使用您需要的 ARN 值。回填請求的配對建構票證 (MatchmakingTicket) 在處理時擁有遊戲工作階段 ID；新配對建構請求的票證不會取得新遊戲工作階段 ID，直到進行配對為止；遊戲工作階段 ID 的存在可讓您了解新配對票證和回填票證之間的差異。
   - 玩家資料 — 在您回填的遊戲工作階段中包含所有目前玩家的玩家資訊 (Player)。此資訊可讓配對建構器為目前在遊戲工作階段中的玩家找到最可能的玩家配對。您必須包含每個玩家的團隊成員資格。如果您未使用回填，請勿指定團隊。如果您的遊戲伺服器已準確報告玩家連線狀態，您應該要能取得此資料，如下所示：

     1. 呼叫含遊戲工作階段 ID 的 DescribePlayerSessions() 來探索目前正在連接到遊戲工作階段的所有玩家。每個玩家工作階段皆包含玩家 ID。您可以新增狀態篩選以只擷取作用中玩家工作階段。

2. 從遊戲工作階段物件 (GameSession)、MatchmakerData 屬性 (請參閱 關於配對建構器資料) 提取玩家資料。使用在之前步驟中取得的玩家 ID，以僅取得目前連線玩家的資料。由於在玩家中途退出時資料建構器資料不會更新，您只需要為目前的玩家擷取資料。
3. 對於玩家延遲，如果配對建構器呼叫延遲資料，從所有目前玩家收集新延遲值，並將其包含在 Player 物件。如果將延遲資料省略和配對建構器有延遲規則，則請求無法成功相配。回填請求只需要遊戲工作階段目前所在之區域的延遲資料。您可以從 GameSession 物件的 GameSessionId 屬性取得遊戲工作階段區域，此值是包含在該區域的 ARN。

3. 追蹤回填請求的狀態。新增程式碼以監聽配對建構票證狀態更新。您可以使用事件通知 (偏好) 或輪詢，透過機制設定來追蹤新配對建構請求的票證 (請參閱 追蹤配對事件)。雖然您不需要觸發玩家接受活動與回填請求，玩家資訊即會在遊戲伺服器上進行更新，您仍需要監控票證狀態來處理請求失敗和重新提交。

   配對建構器可隨時處理來自遊戲工作階段的配對回填請求。如果您需要取消請求，請呼叫 StopMatchmaking。如果您需要變更請求，請呼叫 StopMatchmaking 接著提交更新請求。

   一旦配對回填請求成功，您的遊戲伺服器會收到更新的 GameSession 物件並處理所需的任務以將新玩家加入遊戲工作階段。請參閱 更新遊戲伺服器上的配對資料 了解更多資訊。

更新遊戲伺服器上的配對資料

無論您在遊戲中初始化配對回填請求的方式為何，遊戲伺服器必須能夠處理因配對回填請求，Amazon GameLift Servers 所提供的遊戲工作階段更新。

當 Amazon GameLift Servers完成配對回填請求，無論是否成功，都會使用回呼函數 來呼叫您的遊戲伺服器onUpdateGameSession。此呼叫有三個輸入參數：配對回填票證 ID、狀態訊息，以及包含最新配對資料的 GameSession 物件 (包括玩家資訊)。您需要在遊戲伺服器整合過程中將以下程式碼新增到遊戲伺服器：

1. 實作 onUpdateGameSession 函式。此函式必須能夠處理以下狀態訊息 (updateReason)：
   - MATCHMAKING_DATA_UPDATED – 新玩家已成功符合遊戲工作階段。GameSession 物件包含更新的配對建構資料 (包含有關現有玩家和新配對玩家的玩家資料)。
   - BACKFILL_FAILED – 由於內部錯誤，配對回填嘗試失敗。GameSession 物件保持不變。
   - BACKFILL_TIMED_OUT – 配對建構器無法在時間限制內找到回填配對。GameSession 物件保持不變。
   - BACKFILL_CANCELLED – 對 StopMatchmaking （用戶端） 或 StopMatchBackfill （伺服器） 的呼叫取消配對回填請求。GameSession 物件保持不變。

2. 對於成功回填配對，使用更新的配對建構器資料以在新玩家連接到遊戲工作階段時能處理他們。您至少需要為新玩家使用團隊指派，以及其他所需的玩家屬性讓玩家在遊戲中得以開始。
3. 在遊戲伺服器對伺服器 SDK 動作 ProcessReady() 的呼叫中，新增回onUpdateGameSession呼方法名稱做為程序參數。

FlexMatch 的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 的客戶，您將能從資料中心和網路架構中獲益，這些都是專為最重視安全的組織而設計的。

安全性是 AWS 與您之間的共同責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 Cloud AWS 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。在AWS 合規計劃中，第三方稽核人員會定期測試和驗證我們的安全有效性。若要了解適用於 的合規計劃Amazon GameLift Servers，請參閱AWS 合規計劃範圍內的服務。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須負責其他因素，包括資料的敏感度、公司的要求，以及適用的 l AWS 和法規。

如需 的相關安全性資訊Amazon GameLift Servers，包括 FlexMatch，請參閱 中的安全性Amazon GameLift Servers。本文件有助於您了解如何在使用 Amazon GameLift Servers 時套用共同責任模型。這些主題說明如何設定 Amazon GameLift Servers 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 Amazon GameLift Servers 資源。

Amazon GameLift Servers參考FlexMatch

本節包含與 Amazon GameLift Servers FlexMatch 配對的參考文件。

主題

- Amazon GameLift Servers FlexMatch API 參考 (AWS SDK)
- FlexMatch 規則語言
- FlexMatch 配對事件

Amazon GameLift Servers FlexMatch API 參考 (AWS SDK)

本主題提供 的任務型 API Amazon GameLift Servers 操作清單FlexMatch。Amazon GameLift Servers FlexMatch 服務 API 會封裝在 `aws.gamelift` 命名空間的 AWS SDK 中。下載 AWS SDK 或檢視 Amazon GameLift Servers API 參考文件。

Amazon GameLift Servers FlexMatch 提供配對服務，以搭配Amazon GameLift Servers託管解決方案託管的遊戲（包括自訂遊戲伺服器或 的受管託管Amazon GameLift Servers Realtime，以及搭配在 Amazon EC2 上託管Amazon GameLift Servers FleetIQ)，以及其他託管系統，例如peer-to-peer、內部部署或雲端運算基本概念。如需其他Amazon GameLift Servers託管選項的詳細資訊，請參閱 Amazon GameLift Servers開發人員指南。

主題

- 設定配對規則和程序
- 為玩家或玩家請求配對
- 可用的程式設計語言

設定配對規則和程序

呼叫這些操作來建立FlexMatch配對建構器、設定遊戲的配對程序，以及定義一組自訂規則來建立配對和團隊。

配對組態

- CreateMatchmakingConfiguration – 建立配對組態，其中包含評估玩家群組和建立玩家團隊的指示。使用 Amazon GameLift Servers 託管時，也請指定如何為配對建立新的遊戲工作階段。
- DescribeMatchmakingConfigurations – 擷取定義 Amazon GameLift Servers區域的配對組態。

- UpdateMatchmakingConfiguration – 變更配對組態的設定。 佇列。
- DeleteMatchmakingConfiguration – 從區域移除配對組態。

配對規則集

- CreateMatchmakingRuleSet – 建立一組規則，以便在搜尋玩家配對時使用。
- DescribeMatchmakingRuleSets – 擷取 Amazon GameLift Servers區域中定義的配對規則集。
- ValidateMatchmakingRuleSet – 驗證一組配對規則的語法。
- DeleteMatchmakingRuleSet – 從區域移除配對規則集。

為玩家或玩家請求配對

您可以從遊戲用戶端服務呼叫以下操作，進而管理玩家配對請求。

- StartMatchmaking – 為想要在相同配對中玩遊戲的玩家或群組請求配對。
- DescribeMatchmaking – 取得配對請求的詳細資訊，包括狀態。
- AcceptMatch – 對於需要玩家接受的配對，請在玩家接受提議的配對Amazon GameLift Servers時通知 。
- StopMatchmaking – 取消配對請求。
- StartMatchBackfill – 請求額外的玩家配對以填入現有遊戲工作階段中的空位。

可用的程式設計語言

支援 的 AWS SDK Amazon GameLift Servers 提供下列語言。如需開發環境支援的相關資訊，請參閱每種語言的文件。

- C++ (SDK 文件) (Amazon GameLift Servers)
- Java (SDK 文件) (Amazon GameLift Servers)
- .NET (SDK 文件) (Amazon GameLift Servers)
- Go (SDK 文件) (Amazon GameLift Servers)
- Python (SDK 文件) (Amazon GameLift Servers)
- Ruby (SDK 文件) (Amazon GameLift Servers)
- PHP (SDK 文件) (Amazon GameLift Servers)
- JavaScript/Node.js (SDK 文件) (Amazon GameLift Servers)

FlexMatch 規則語言

本節中的參考主題描述用於建置配對規則以搭配 Amazon GameLift Servers 使用的語法和語意 FlexMatch。如需撰寫配對規則和規則集的詳細說明，請參閱 建置FlexMatch規則集。

主題

- FlexMatch 規則集結構描述
- FlexMatch 規則集屬性定義
- FlexMatch 規則類型
- FlexMatch 屬性表達式

FlexMatch 規則集結構描述

FlexMatch 規則集會將標準結構描述用於小型配對和大型配對規則。如需每個區段的詳細說明，請參閱 FlexMatch 規則集屬性定義。

小型配對的規則集結構描述

下列結構描述會記錄規則集的所有可能屬性和允許值，用於最多 40 名玩家的建置配對。

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{
         "name": "string,
         "type": <"string", "number", "string_list", "string_number_map">,
         "default": "string"
    }],
    "algorithm": {
        "strategy": "exhaustiveSearch",
        "batchingPreference": <"random", "sorted">,
        "sortByAttributes": [ "string" ],
        "expansionAgeSelection": <"newest", "oldest">,
        "backfillPriority": <"normal", "low", "high">
    },
    "teams": [{
        "name": "string",
        "maxPlayers": number,
        "minPlayers": number,
        "quantity": integer
```

```
    }],
    "rules": [{
        "type": "distance",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "maxDistance": number,
        "minDistance": number,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "comparison",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "operation": <"<", "<=", "=", "!=", ">", ">=">,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "collection",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "operation": <"intersection", "contains", "reference_intersection_count">,
        "maxCount": number,
        "minCount": number,
        "partyAggregation": <"union", "intersection">
        },{
        "type": "latency",
        "name": "string",
        "description": "string",
        "maxLatency": number,
        "maxDistance": number,
        "distanceReference": number,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "distanceSort",
        "name": "string",
        "description": "string",
        "sortDirection": <"ascending", "descending">,
        "sortAttribute": "string",
        "mapKey": <"minValue", "maxValue">,
        "partyAggregation": <"avg", "min", "max">
```

```
        },{
        "type": "absoluteSort",
        "name": "string",
        "description": "string",
        "sortDirection": <"ascending", "descending">,
        "sortAttribute": "string",
        "mapKey": <"minValue", "maxValue">,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "compound",
        "name": "string",
        "description": "string",
        "statement": "string"
        }
    }],
    "expansions": [{
        "target": "string",
        "steps": [{
            "waitTimeSeconds": number,
            "value": number
        }, {
            "waitTimeSeconds": number,
            "value": number
        }]
    }]
}
```

大型比對的規則集結構描述

下列結構描述會記錄規則集的所有可能屬性和允許值，用於建置超過 40 名玩家的配對。如果規則集中所有團隊`maxPlayers`的值總計超過 40，則 FlexMatch會處理根據大型比對準則使用此規則集的比對請求。

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{
         "name": "string,
         "type": <"string", "number", "string_list", "string_number_map">,
         "default": "string"
    }],
    "algorithm": {
        "strategy": "balanced",
```

```
        "batchingPreference": <"largestPopulation", "fastestRegion">,
        "balancedAttribute": "string",
        "expansionAgeSelection": <"newest", "oldest">,
        "backfillPriority": <"normal", "low", "high">
    },
    "teams": [{
        "name": "string",
        "maxPlayers": number,
        "minPlayers": number,
        "quantity": integer
    }],
    "rules": [{
        "name": "string",
        "type": "latency",
        "description": "string",
        "maxLatency": number,
        "partyAggregation": <"avg", "min", "max">
    }, {
        "name": "string",
        "type": "batchDistance",
        "batchAttribute": "string",
        "maxDistance": number
    }],
    "expansions": [{
        "target": "string",
        "steps": [{
            "waitTimeSeconds": number,
            "value": number
        }, {
            "waitTimeSeconds": number,
            "value": number
        }]
    }]
}
```

FlexMatch 規則集屬性定義

本節定義規則集結構描述中的每個屬性。如需建立規則集的其他說明，請參閱 建置FlexMatch規則集。

name

規則集的描述性標籤。此值與指派給 Amazon GameLift Servers MatchmakingRuleSet 資源的名稱無關。此值會包含在描述已完成配對的配對資料中，但不會用於任何Amazon GameLift Servers程序。

允許的值：字串

是否為必要？ 否

ruleLanguageVersion

正在使用的 FlexMatch 屬性表達式語言版本。

允許的值："1.0"

是否為必要？ 是

playerAttributes

包含在配對請求中的玩家資料集合，用於配對程序。您也可以在此處宣告屬性，讓玩家資料包含在傳遞至遊戲伺服器的配對資料中，即使資料未用於配對程序。

是否為必要？ 否

name

配對建構器要使用的玩家屬性的唯一名稱。此名稱必須符合配對請求中參考的玩家屬性名稱。

允許的值：字串

是否為必要？ 是

type

玩家屬性值的資料類型。

允許的值："string"、"number"、"string_list"、"string_number_map"

是否為必要？ 是

default

當配對請求未提供玩家時要使用的預設值。

允許的值：播放器屬性允許的任何值。

是否為必要？ 否

algorithm

自訂配對程序的選用組態設定。

是否為必要？ 否

strategy

建置相符項目時要使用的 方法。如果未設定此屬性，則預設行為為「exhaustiveSearch」。

允許的值：

- 「exhaustiveSearch」 – 標準比對方法。 會根據一組自訂FlexMatch比對規則，評估集區中的其他票證，在批次中形成最舊票證的比對。此策略用於 40 名或更少玩家的配對。使用此策略時， `batchingPreference`應設定為「隨機」或「排序」。
- "balanced" – 最佳化以快速形成大型配對的方法。此策略僅用於 41 到 200 名玩家的配對。它透過預先排序票證集區、建立潛在配對並將玩家指派給隊伍，然後使用指定的玩家屬性平衡配對中的每個隊伍來形成配對。例如，此策略可用來平衡配對中所有團隊的平均技能水準。使用此策略時， `balancedAttribute` 必須設定 ，且 `batchingPreference`應設定為「largestPopulation」或「fastestRegion」。此策略無法辨識大多數自訂規則類型。

是否為必要？ 是

batchingPreference

分組配對建置票證之前要使用的預先排序方法。預先排序票證集區會導致根據特定特徵將票證批次在一起，這往往會增加最終配對中玩家之間的一致性。

允許的值：

- "random" – 僅適用於 `strategy` = "exhaustiveSearch"。未完成預先排序；集區中的票證會隨機批次處理。這是詳盡搜尋策略的預設行為。
- "sorted" – 僅適用於 `strategy` = "exhaustiveSearch"。票證集區會根據 中列出的玩家屬性預先排序`sortbyAttributes`。
- "largestPopulation" – 僅適用於 `strategy` = "balanced"。票證集區會依玩家回報可接受延遲層級的區域預先排序。這是平衡策略的預設行為。
- "fastestRegion" – 僅適用於 `strategy` = "balanced"。票證集區會依玩家回報其最低延遲等級的區域預先排序。產生的配對需要更長的時間才能完成，但所有玩家的延遲往往很低。

是否為必要？ 是

balancedAttribute

使用平衡策略建置大型配對時要使用的玩家屬性名稱。

允許的值：playerAttributes在 中宣告且 type = "number" 的任何屬性。

是否為必要？ 是，如果 strategy = "balanced"。

sortByAttributes

在批次處理之前預先排序票證集區時要使用的玩家屬性清單。只有在使用詳盡的搜尋策略預先排序時，才會使用此屬性。屬性清單的順序決定排序順序。 使用 Alpha 和數值FlexMatch的標準排序慣例。

允許的值：在 中宣告的任何屬性playerAttributes。

是否為必要？ 是，如果 batchingPreference = "sorted"。

backfillPriority

符合回填票證的優先順序方法。此屬性會決定何時在批次中FlexMatch處理回填票證。只有在使用詳盡的搜尋策略預先排序時才會使用它。如果未設定此屬性，則預設行為為「正常」。

允許的值：

- "normal" – 形成相符項目時，不會考慮票證的請求類型 （回填或新相符項目）。
- "high" – 票證批次會依請求類型 （然後依年齡排序） 排序，並FlexMatch嘗試先比對回填票證。
- 「低」：票證批次會依請求類型 （然後依年齡排序） 排序，並FlexMatch嘗試先比對未回填的票證。

是否為必要？ 否

expansionAgeSelection

計算相符規則擴展的等待時間的方法。如果在經過一定的時間後仍未完成配對，則擴展會用來放寬配對要求。等待時間是根據已在部分填充配對中票證的存留期計算。如果未設定此屬性，則預設行為為「最新」。

允許的值：

- 「最新」：擴展等待時間是根據部分完成配對中具有最新建立時間戳記的票證計算。擴展觸發速度較慢，因為較新的票證可以重新啟動等待時鐘。
- 「最舊」：擴充等待時間是根據相符項目中建立時間戳記最舊的票證來計算。擴展通常會更快觸發。

是否為必要？ 否

teams

配對中團隊的組態。為每個團隊提供團隊名稱和大小範圍。規則集必須至少定義一個團隊。

name

團隊的唯一名稱。團隊名稱可以在規則和擴展中參考。在成功配對時，玩家會依配對資料中的隊伍名稱指派。

允許的值：字串

是否為必要？ 是

maxPlayers

可指派給團隊的玩家數量上限。

允許的值：數字

是否為必要？ 是

minPlayers

在配對可行之前，必須指派給隊伍的玩家人數下限。

允許的值：數字

是否為必要？ 是

quantity

要在配對中建立的此類型團隊數量。數量大於 1 的團隊會以附加數字 ("Red_1"、"Red_2" 等) 指定。如果未設定此屬性，預設值為 "1"。

允許的值：數字

是否為必要？ 否

rules

定義如何評估玩家配對的規則陳述式集合。

是否為必要？ 否

name

規則的唯一名稱。規則集中的所有規則都必須具有唯一的名稱。在追蹤規則相關活動的事件日誌和指標中，會參考規則名稱。

允許的值：字串

是否為必要？ 是

description

規則的文字描述。此資訊可用於識別規則的目的。它不會用於配對程序。

允許的值：字串

是否為必要？ 否

type

規則陳述式的類型。每個規則類型都有必須設定的其他屬性。如需每個規則類型的結構和使用詳細資訊，請參閱 FlexMatch 規則類型。

允許的值：

- "absoluteSort" – 根據指定的玩家屬性是否與批次中最舊的票證進行比較，使用明確排序方法來排序批次中的票證。
- 「集合」 – 評估集合中的值，例如集合的玩家屬性，或多個玩家的一組值。
- "comparison" – 比較兩個值。
- "compound" – 使用規則集中其他規則的邏輯組合來定義複合配對規則。僅支援 40 名或更少玩家的配對。
- "distance" – 測量數值之間的距離。
- "batchDistance" – 測量屬性值之間的差異，並使用它來分組相符請求。
- "distanceSort" – 根據指定玩家屬性與批次中最舊的票證的比較，使用明確排序方法來排序批次中的票證。
- "latency" – 評估針對配對請求報告的區域延遲資料。

是否為必要？ 是

expansions

當無法完成配對時，隨時間放寬配對要求的規則。將擴展設定為一系列逐步套用的步驟，以便更容易找到相符項目。根據預設， 會根據新增至配對的最新票證存留期來FlexMatch計算等待時間。您可以使用演算法屬性 來變更計算擴展等待時間的方式`expansionAgeSelection`。

擴展等待時間是絕對值，因此每個步驟的等待時間應該比上一個步驟長。例如，若要排程漸進一系列的擴展，您可以使用 30 秒、40 秒和 50 秒的等待時間。等待時間不能超過配對請求允許的最長時間，該請求是在配對組態中設定。

是否為必要？ 否

target

要放寬的規則集元素。您可以放寬團隊大小屬性或任何規則陳述式屬性。語法為「<component name>【<rule/team name>】.<property name>」。例如，若要變更團隊大小下限：`teams[Red, Yellow].minPlayers`。若要在名為 "minSkill" 的比較規則陳述式中變更最低技能需求：`rules[minSkill].referenceValue`。

是否為必要？ 是

steps

waitTimeSeconds

套用目標規則集元素的新值之前等待的時間長度，以秒為單位。

是否為必要？ 是

value

目標規則集元素的新值。

FlexMatch 規則類型

批次距離規則

```
batchDistance
```

批次距離規則會測量兩個屬性值之間的差異。您可以對大型和小型相符項目使用批次距離規則類型。批次距離規則有兩種類型：

- 比較數值屬性值。例如，此類型的批次距離規則可能需要配對中的所有玩家彼此在兩個技能層級內。對於此類型，定義所有票證`batchAttribute`的 之間的最大距離。
- 比較字串屬性值。例如，此類型的批次距離規則可能需要配對中的所有玩家請求相同的遊戲模式。針對此類型，定義 FlexMatch 用來形成批次`batchAttribute`的值。

批次距離規則屬性

- **batchAttribute** – 用來形成批次的玩家屬性值。
- **maxDistance** – 成功配對的最大距離值。用來比較數值屬性。

- **partyAggregation** – 決定 如何處理具有多個玩家 （ 各方 ） 之FlexMatch票證的值。有效選項包括門票玩家的最小值 (min)、最大值 (max) 和平均值 (avg)。預設值為 avg。

Example

範例

```
{
  "name":"SimilarSkillRatings",
  "description":"All players must have similar skill ratings",
  "type":"batchDistance",
  "batchAttribute":"SkillRating",
  "maxDistance":"500"
}
```

```
{
  "name":"SameGameMode",
  "description":"All players must have the same game mode",
  "type":"batchDistance",
  "batchAttribute":"GameMode"
}
```

比較規則

```
comparison
```

比較規則會將玩家屬性值與另一個值進行比較。比較規則有兩種類型：

- 與參考值比較。例如，此類型的比較規則可能需要符合的玩家具有特定的技能水準或更高。對於此類型，請指定玩家屬性、參考值和比較操作。
- 比較玩家。例如，此類型的比較規則可能需要配對中的所有玩家使用不同的字元。對於此類型，請指定玩家屬性和等於 (=) 或不等於 (!=) 比較操作。請勿指定參考值。

> **Note**
>
> 批次距離規則在比較玩家屬性方面更有效率。若要減少配對延遲，請盡可能使用批次距離規則。

比較規則屬性

- **measurements** – 要比較的玩家屬性值。
- **referenceValue** – 要針對潛在相符項目比較測量值與 的值。
- **operation** – 決定如何比較測量值與參考值的值。有效操作包括：<、<=、=、!=、>、>=。
- **partyAggregation** – 決定 如何處理具有多個玩家 （ 各方 ） 之FlexMatch票證的值。有效選項包括門票玩家的最小值 (min)、最大值 (max) 和平均值 (avg)。預設值為 avg。

距離規則

```
distance
```

距離規則會測量兩個數值之間的差異，例如玩家技能等級之間的距離。例如，距離規則可能需要所有玩家都玩遊戲至少 30 小時。

> **Note**
>
> 批次距離規則在比較玩家屬性方面更有效率。若要減少配對延遲，請盡可能使用批次距離規則。

距離規則屬性

- **measurements** – 要測量距離的玩家屬性值。這必須是具有數值的屬性。
- **referenceValue** – 測量潛在配對距離的數值。
- **minDistance/maxDistance** – 成功配對的最小或最大距離值。
- **partyAggregation** – 決定 如何處理具有多個玩家 （ 各方 ） 之FlexMatch票證的值。有效選項包括門票玩家的最小值 (min)、最大值 (max) 和平均值 (avg)。預設值為 avg。

集合規則

```
collection
```

集合規則會將一組玩家屬性值與批次中其他玩家的值或參考值進行比較。集合可以包含多個玩家的屬性值、做為字串清單的玩家屬性，或兩者。例如，集合規則可能會查看隊伍中玩家選擇的字元。然後，規則可能要求團隊至少擁有一個特定字元。

集合規則屬性

- **measurements** – 要比較的玩家屬性值集合。屬性值必須是字串清單。
- **referenceValue** – 用來比較潛在比對之測量的值 （或值集合）。
- **operation** – 決定如何比較測量集合的值。有效操作包括下列項目：
  - intersection – 此操作會測量所有玩家集合中相同的值數量。如需使用交集操作的規則範例，請參閱 範例：使用明確排序來尋找最佳相符項目。
  - contains – 此操作會測量包含指定參考值的玩家屬性集合數目。如需使用 包含 操作的規則範例，請參閱 範例：設定團隊層級需求和延遲限制。
  - reference_intersection_count – 此操作會測量玩家屬性集合中符合參考值集合中項目的項目數量。您可以使用此操作來比較多個不同的玩家屬性。如需比較多個玩家屬性集合的規則範例，請參閱 範例：尋找跨多個玩家屬性的交集。
- **minCount/maxCount** – 成功配對的最小或最大計數值。
- **partyAggregation** – 決定 如何處理具有多個玩家 （各方） 之FlexMatch票證的值。對於此值，您可以使用 union來合併隊伍中所有玩家的玩家屬性。或者，您可以使用 intersection 來使用一方共同的玩家屬性。預設值為 union。

複合規則

```
compound
```

複合規則使用邏輯陳述式來形成 40 名或更少玩家的配對。您可以在單一規則集中使用多個複合規則。使用多個複合規則時，所有複合規則都必須為 true 才能形成相符項目。

您無法使用擴展規則擴展複合規則，但您可以擴展基礎或支援規則。

複合規則屬性

- **statement** – 用來結合個別規則以形成複合規則的邏輯。您在此屬性中指定的規則必須已在規則集的先前定義。您無法在複合batchDistance規則中使用規則。

  此屬性支援下列邏輯運算子：
  - and – 如果兩個提供的引數為 true，表示表達式為 true。
  - or – 如果兩個提供的引數之一為 true，則表達式為 true。
  - not – 反轉表達式中引數的結果。
  - xor – 如果只有一個引數為 true，則表達式為 true。

Example範例

下列範例會根據玩家選取的遊戲模式，比對不同技能等級的玩家。

```
{
    "name": "CompoundRuleExample",
    "type": "compound",
    "statement": "or(and(SeriousPlayers, VeryCloseSkill), and(CasualPlayers,
 SomewhatCloseSkill))"
}
```

延遲規則

```
latency
```

延遲規則會測量每個位置的玩家延遲。延遲規則會忽略延遲高於最大值的任何位置。玩家在至少一個位置的延遲值必須低於最大值，延遲規則才能接受它們。您可以指定 `maxLatency` 屬性，將此規則類型與大型相符項目搭配使用。

延遲規則屬性

- **`maxLatency`** – 位置可接受的延遲上限值。如果票證沒有延遲低於最大值的位置，則票證不符合延遲規則。
- **`maxDistance`** – 每個票證的延遲與距離參考值之間的最大值。
- **`distanceReference`** – 要比較票證延遲的延遲值。在距離參考值最大距離內的票證會導致成功配對。有效選項包括最低 (`min`) 和平均 (`avg`) 玩家延遲值。
- **`partyAggregation`** – 決定 如何處理具有多個玩家 （ 各方 ） 之FlexMatch票證的值。有效選項包括門票玩家的最小值 (`min`)、最大值 (`max`) 和平均值 (`avg`)。預設值為 `avg`。

> **Note**
>
> 佇列可以將遊戲工作階段放置在不符合延遲規則的區域。如需佇列延遲政策的詳細資訊，請參閱建立玩家延遲政策。

絕對排序規則

```
absoluteSort
```

相較於新增至批次的第一個票證，絕對排序規則會根據指定的玩家屬性來排序一批配對票證。

絕對排序規則屬性

- **sortDirection** – 排序配對票證的順序。有效選項包括 ascending和 descending。
- **sortAttribute** – 排序票證的玩家屬性。
- **mapKey** – 排序玩家屬性的選項，如果是地圖的話。有效的選項包含：
  - minValue – 值最低的金鑰為第一個。
  - maxValue – 具有最高值的金鑰優先。
- **partyAggregation** – 決定 如何處理具有多個玩家 （ 各方 ） 之FlexMatch票證的值。有效選項包括玩家屬性下限 (min)、玩家屬性上限 (max)，以及隊伍中玩家所有玩家屬性的平均值 (avg)。預設值為 avg。

Example

範例

下列範例規則會依技能層級排序玩家，並平均各方的技能層級。

```
{
    "name":"AbsoluteSortExample",
    "type":"absoluteSort",
    "sortDirection":"ascending",
    "sortAttribute":"skill",
    "partyAggregation":"avg"
}
```

距離排序規則

```
distanceSort
```

距離排序規則會根據指定玩家屬性與新增至批次的第一個票證的距離，來排序一批配對票證。

距離排序規則屬性

- **sortDirection** – 排序配對票證的方向。有效選項包括 ascending和 descending。
- **sortAttribute** – 排序票證的玩家屬性。
- **mapKey** – 排序玩家屬性的選項，如果是地圖的話。有效的選項包含：

  - `minValue` – 針對新增至批次的第一個票證，尋找值最低的索引鍵。
  - `maxValue` – 針對新增至批次的第一個票證，尋找具有最高值的金鑰。
- **`partyAggregation`** – 決定 如何處理具有多個玩家 （ 各方 ） 之FlexMatch票證的值。有效選項包括門票玩家的最小值 (`min`)、最大值 (`max`) 和平均值 (`avg`)。預設值為 `avg`。

FlexMatch 屬性表達式

屬性表達式可用來定義特定配對相關屬性。它們可讓您在定義屬性值時使用計算和邏輯。屬性表達式通常會產生兩種形式之一：

- 個別玩家資料。
- 計算個別玩家資料的集合。

常見的配對屬性表達式

屬性表達式可識別玩家、團隊或配對的特定值。以下部分表達式說明了如何辨識團隊和玩家：

目標	Input	意義	Output
辨識配對中的特定團隊：	`teams[red]`	紅隊	團隊
若要識別配對中的一組特定團隊：	`teams[red,blue]`	紅隊和藍隊	List<Team>
辨識配對中的所有團隊：	`teams[*]`	所有團隊	List<Team>
辨識特定團隊中的玩家：	`team[red] .players`	紅隊中的玩家	List<Player>
若要在配對中識別一組特定隊伍中的玩家：	`team[red, blue].players`	配對的玩家，依照團隊分組	List<List<Player>>
辨識配對中的玩家：	`team[*].players`	配對的玩家，依照團隊分組	List<List<Player>>

屬性表達式範例

下表說明一些以先前範例為基礎的屬性表達式：

表達式	意義	結果類型
`teams[red].players[playerId]`	紅隊所有玩家的玩家 ID	List<string>
`teams[red].players.attributes[skill]`	紅隊所有玩家的「技能」屬性	List<number>
`teams[red,blue].players.attributes[skill]`	紅隊和藍隊上所有玩家的「技能」屬性，依隊伍分組	List<List<number>>
`teams[*].players.attributes[skill]`	配對的所有玩家的「技能」屬性 (依照團隊分組)	List<List<number>>

屬性表達式彙總

屬性表達式可用於使用以下函式或組合函式來整合團隊資料：

聚合	Input	意義	Output
`min`	List<number>	取得列表中所有數字的最小值。	number
`max`	List<number>	取得列表中所有數字的最大值。	number
`avg`	List<number>	取得列表中所有數字的平均值。	number
`median`	List<number>	取得列表中所有數字的中間值。	number

聚合	Input	意義	Output
`sum`	List<number>	取得列表中所有數字的總和。	number
`count`	List<?>	取得列表中所有元素的數量。	number
`stddev`	List<number>	取得列表中所有數字的標準差。	number
`flatten`	List<List<?>>	將嵌套清單的集合變成包含所有元素的單一清單。	List<?>
`set_intersection`	List<List<string>>	取得在集合的所有字串清單中找到的字串清單。	List<string>
以上全部	List<List<?>>	對嵌套列表的所有操作會對每個子列表執行一次以產生結果列表。	List<?>

下表說明使用彙總函式的部分有效屬性表達式：

表達式	意義	結果類型
flatten(teams[*].players.attributes[skill])	配對的所有玩家的「技能」屬性 (未分組)	List<number>
avg(teams[red].players.attributes[skill])	紅隊玩家的平均技能	number
avg(teams【*】.players.attributes【skill】)	配對中的每個團隊的平均技能	List<number>
avg(flatten(teams[*].players.attributes[skill]))	配對中的所有玩家的平均技能級別。此表	number

表達式	意義	結果類型
	達式取得玩家技能的展平列表，然後計算它們的平均值。	
count(teams[red].players)	紅隊玩家的數量	number
count (teams[*].players)	配對中的每個團隊的玩家數量	List<number>
max(avg(teams[*].players.attributes[skill]))	配對中的最高團隊技能級別	number

FlexMatch 配對事件

Amazon GameLift Servers FlexMatch 在處理每個配對票證時發出事件。您可以將這些事件發佈至 Amazon SNS 主題，如中所述設定FlexMatch事件通知。這些事件也會近乎即時地傳送至 Amazon CloudWatch Events。

本主題說明FlexMatch事件的結構，並提供每個事件類型的範例。如需配對票證狀態的詳細資訊，請參閱 Amazon GameLift Servers API 參考中的MatchmakingTicket。

主題

- MatchmakingSearching
- PotentialMatchCreated
- AcceptMatch
- AcceptMatchCompleted
- MatchmakingSucceeded
- MatchmakingTimedOut
- MatchmakingCancelled
- MatchmakingFailed

MatchmakingSearching

已輸入配對的票證。包括新請求和屬於已失敗建議配對的請求。

資源：ConfigurationArn

詳細資訊：type、tickets、estimatedWaitMillis、gameSessionInfo

範例

```
{
  "version": "0",
  "id": "cc3d3ebe-1d90-48f8-b268-c96655b8f013",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T21:15:36.421Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T21:15:35.676Z",
        "players": [
          {
            "playerId": "player-1"
          }
        ]
      }
    ],
    "estimatedWaitMillis": "NOT_AVAILABLE",
    "type": "MatchmakingSearching",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1"
        }
      ]
    }
  }
}
```

PotentialMatchCreated

已建立的潛在配對。這是發出給所有潛在的新配對，無論是否需要接受。

資源：ConfigurationArn

詳細資
訊：type、tickets、acceptanceTimeout、acceptanceRequired、ruleEvaluationMetrics、gameSessionInfo、

範例

```
{
  "version": "0",
  "id": "fce8633f-aea3-45bc-aeba-99d639cad2d4",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T21:17:41.178Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T21:15:35.676Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-08T21:17:40.657Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue"
          }
```

```
        ]
      }
    ],
    "acceptanceTimeout": 600,
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 3,
        "failedCount": 0
      }
    ],
    "acceptanceRequired": true,
    "type": "PotentialMatchCreated",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue"
        }
      ]
    },
    "matchId": "3faf26ac-f06e-43e5-8d86-08feff26f692"
  }
}
```

AcceptMatch

玩家已接受潛在配對。此事件包含目前各場配對中每個玩家的接受狀態。缺失資料代表玩家尚未呼叫 AcceptMatch。

資源：ConfigurationArn

詳細資訊：type、tickets、matchId、gameSessionInfo

範例

```
{
  "version": "0",
  "id": "b3f76d66-c8e5-416a-aa4c-aa1278153edc",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:04:42.660Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T20:04:16.637Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue",
            "accepted": false
```

```
          }
        ]
      }
    ],
    "type": "AcceptMatch",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue",
          "accepted": false
        }
      ]
    },
    "matchId": "848b5f1f-0460-488e-8631-2960934d13e5"
  }
}
```

AcceptMatchCompleted

配對接受狀態會因玩家接受、拒絕或接受逾時而完成。

資源：ConfigurationArn

詳細資訊：type、tickets、acceptance、matchId、gameSessionInfo

範例

```
{
  "version": "0",
  "id": "b1990d3d-f737-4d6c-b150-af5ace8c35d3",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-08T20:43:14.621Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
```

```
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-08T20:30:40.972Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-08T20:33:14.111Z",
        "players": [
          {
            "playerId": "player-2",
            "team": "blue"
          }
        ]
      }
    ],
    "acceptance": "TimedOut",
    "type": "AcceptMatchCompleted",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "team": "blue"
        }
      ]
    },
    "matchId": "a0d9bd24-4695-4f12-876f-ea6386dd6dce"
  }
}
```

MatchmakingSucceeded

已成功完成配對，且已建立遊戲工作階段。

資源：ConfigurationArn

詳細資訊：type、tickets、matchId、gameSessionInfo

範例

```
{
  "version": "0",
  "id": "5ccb6523-0566-412d-b63c-1569e00d023d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T19:59:09.159Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T19:58:59.277Z",
        "players": [
          {
            "playerId": "player-1",
            "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
            "team": "red"
          }
        ]
      },
      {
        "ticketId": "ticket-2",
        "startTime": "2017-08-09T19:59:08.663Z",
        "players": [
          {
            "playerId": "player-2",
            "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
            "team": "blue"
          }
```

```
        ]
      }
    ],
    "type": "MatchmakingSucceeded",
    "gameSessionInfo": {
      "gameSessionArn": "arn:aws:gamelift:us-west-2:123456789012:gamesession/836cf48d-
bcb0-4a2c-bec1-9c456541352a",
      "ipAddress": "192.168.1.1",
      "port": 10777,
      "players": [
        {
          "playerId": "player-1",
          "playerSessionId": "psess-6e7c13cf-10d6-4756-a53f-db7de782ed67",
          "team": "red"
        },
        {
          "playerId": "player-2",
          "playerSessionId": "psess-786b342f-9c94-44eb-bb9e-c1de46c472ce",
          "team": "blue"
        }
      ]
    },
    "matchId": "c0ec1a54-7fec-4b55-8583-76d67adb7754"
  }
}
```

MatchmakingTimedOut

配對票證因逾時而失敗。

資源：ConfigurationArn

詳細資訊：type、tickets、ruleEvaluationMetrics、message、matchId、gameSessionInfo

範例

```
{
  "version": "0",
  "id": "fe528a7d-46ad-4bdc-96cb-b094b5f6bf56",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:11:35.598Z",
```

```
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "reason": "TimedOut",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T20:01:35.305Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      }
    ],
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 3,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 3,
        "failedCount": 0
      }
    ],
    "type": "MatchmakingTimedOut",
    "message": "Removed from matchmaking due to timing out.",
    "gameSessionInfo": {
```

```
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
      ]
    }
  }
}
```

MatchmakingCancelled

由於 StopMatchmaking API 呼叫，配對票證已取消。

資源：ConfigurationArn

詳細資訊：type、tickets、ruleEvaluationMetrics、message、matchId、gameSessionInfo

範例

```
{
  "version": "0",
  "id": "8d6f84da-5e15-4741-8d5c-5ac99091c27f",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-09T20:00:07.843Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "reason": "Cancelled",
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-09T19:59:26.118Z",
        "players": [
          {
            "playerId": "player-1"
          }
        ]
```

```
      }
    ],
    "ruleEvaluationMetrics": [
      {
        "ruleName": "EvenSkill",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "EvenTeams",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "FastConnection",
        "passedCount": 0,
        "failedCount": 0
      },
      {
        "ruleName": "NoobSegregation",
        "passedCount": 0,
        "failedCount": 0
      }
    ],
    "type": "MatchmakingCancelled",
    "message": "Cancelled by request.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1"
        }
      ]
    }
  }
}
```

MatchmakingFailed

配對票證遭遇錯誤。可能是因為遊戲工作階段佇列無法存取或有內部錯誤。

資源：ConfigurationArn

詳細資訊：type、tickets、ruleEvaluationMetrics、message、matchId、gameSessionInfo

範例

```
{
  "version": "0",
  "id": "025b55a4-41ac-4cf4-89d1-f2b3c6fd8f9d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-16T18:41:09.970Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-16T18:41:02.631Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      }
    ],
    "customEventData": "foo",
    "type": "MatchmakingFailed",
    "reason": "UNEXPECTED_ERROR",
    "message": "An unexpected error was encountered during match placing.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
      ]
    },
    "matchId": "3ea83c13-218b-43a3-936e-135cc570cba7"
  }
}
```

Amazon GameLift Servers 版本備註和 SDK 版本

Amazon GameLift Servers 版本備註提供與服務相關的新Amazon GameLift Servers功能、更新和修正的詳細資訊，包括 FlexMatch 功能。您也可以找到所有 SDKs和外掛程式的Amazon GameLift Servers版本歷史記錄。

- Amazon GameLift Servers SDK 版本
- Amazon GameLift Servers 版本備註

Amazon GameLift Servers 開發人員資源

若要檢視所有Amazon GameLift Servers文件和開發人員資源，請參閱 Amazon GameLift Servers 文件首頁。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。