FlexMatch 開發者指南

Amazon GameLift

版本

Amazon GameLift: FlexMatch 開發者指南

Table of Contents

什麼是 Amazon GameLift FlexMatch？

Amazon GameLift FlexMatch 是一個可定制的多人遊戲配對服務。使用 FlexMatch，您可以建立一組自訂規則來定義多人遊戲對戰的外觀，並決定如何為每場比賽評估和選擇相容玩家。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。

可作 FlexMatch 為獨立配對服務使用，或與 Amazon GameLift 遊戲代管解決方案整合使用。例如，您可以 FlexMatch 將具有 peer-to-peer 架構的遊戲或使用其他雲端運算解決方案的遊戲做為獨立功能來實作。或者，您可 FlexMatch 以添加到您的 Amazon 託 GameLift 管 EC2 託管或現場部署託管與 Amazon GameLift Anywhere。本指南提供有關如何為您的特定情況建立 FlexMatch 配對系統的詳細資訊。

FlexMatch 根據您的遊戲需求，讓您可以靈活地設定配對優先順序。例如，您可以執行下列動作：

- 在比賽速度和質量之間找到平衡。設置比賽規則以快速找到足夠好的比賽，或者讓玩家等待更長時間以找到最佳匹配以獲得最佳球員體驗。
- 根據匹配良好的球員或匹配良好的球隊進行比賽。創建所有玩家都具有相似特徵（例如技能或經驗）的比賽。或者形式匹配，其中每個團隊的組合特徵符合一個共同的標準。
- 優先考慮玩家延遲因素如何進入配對。您是否要為所有玩家設定硬性延遲限制，或者只要比賽中的每個人都有相似的延遲，就可以接受更高的延遲？

> **準備好開始使用了 FlexMatch嗎？**
>
> 如需啟動並執行遊戲的 step-by-step 指引 FlexMatch，請參閱下列主題：
>
> - FlexMatch與亞馬遜GameLift託管集成
> - 用於獨立配對的亞馬遜GameLiftFlexMatch集成

主要 FlexMatch 功能

以下功能適用於所有 FlexMatch 案例，無論您是 FlexMatch作為獨立服務還是搭配 Amazon GameLift 遊戲託管使用。

- 可定制的玩家匹配。設計和構建分房系統，以適應您為玩家提供的所有遊戲模式。建立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，以便為您的遊戲形成絕佳的玩家比賽。

- 基於延遲的匹配。提供玩家延遲資料並建立比賽規則，要求比賽中的玩家擁有類似的回應時間。當您的玩家配對池橫跨多個地理區域時，此功能非常有用。
- Support 最多 200 名玩家的比賽規模。使用為您的遊戲自定義的比賽規則創建最多 40 名玩家的比賽。使用簡化的自定義匹配過程來保持玩家等待時間可管理的匹配過程，創建最多 200 名玩家的比賽。
- 玩家接受度。要求玩家在完成比賽和開始遊戲階段之前選擇加入提議的比賽。使用此功能可啟動您的自訂接受工作流程，並在為比賽進行新的遊戲工作階段 FlexMatch 之前回報玩家的回應。如果並非所有玩家都接受一場比賽，則提議的對戰將失敗，並且接受的玩家會自動返回配對池。
- 玩家派對支持。為想要在同一支球隊中一起比賽的玩家組產生比賽。根據需 FlexMatch 要使用來尋找其他玩家來填寫比賽。
- 可擴展的匹配規則。在經過一定時間後，逐漸放鬆比賽要求，而沒有找到成功的比賽。規則擴展可讓您決定放寬初始比賽規則的位置和時間，以便玩家可以更快地進入可玩的遊戲。
- 匹配回填。在現有遊戲工作階段中填滿空白的玩家插槽，並與新玩家配對。自定義何時以及如何請求新玩家，並使用相同的自定義比賽規則來尋找其他玩家。

FlexMatch 與 Amazon GameLift 託管

FlexMatch 提供下列額外功能，可用於您透過 Amazon 託管的遊戲 GameLift。這包括帶有自定義遊戲服務器或實時服務器的遊戲。

- 遊戲工作階段位置。成功進行配對後，會 FlexMatch 自動向 Amazon 請求新的遊戲工作階段放置位置 GameLift。配對期間產生的資料（包括玩家 ID 和隊伍指派）會提供給遊戲伺服器，以便它可以使用該資訊來開始比賽的遊戲工作階段。 FlexMatch 然後傳回遊戲會話連接信息，以便遊戲客戶端可以加入遊戲。若要將玩家在比賽中所遭受的延遲降到最低，Amazon 的遊戲工作階段配置也 GameLift 可以使用區域性玩家延遲資料 (若有提供)。
- 自動匹配回填。啟用此功能後， FlexMatch 當新的遊戲作業階段以未填滿的玩家欄位開始時，會自動傳送對戰回填要求。您的配對系統會以最少的玩家數量啟動遊戲階段配置程序，然後快速填滿剩餘的欄位。您無法使用自動回填來取代退出配對遊戲工作階段的玩家。

如果您將 Amazon GameLift FleetIQ 與以 Amazon Elastic Compute Cloud (Amazon EC2) 資源託管的遊戲搭配使用，請以獨立服務的形式實 FlexMatch 作。

Amazon 定價 GameLift FlexMatch

Amazon 會依使用期間和傳輸的資料量 GameLift 收取執行個體的費用，以及頻寬的費用。如果您在 Amazon GameLift 服務器上託管遊戲， FlexMatch 用量將包含在 Amazon 的費用中 GameLift。如果您在其他伺服器解決方案上託管遊戲，則 FlexMatch 使用量需另行付費。有關 Amazon 的費用和價格的完整列表 GameLift，請參閱 Amazon GameLift 定價。

有關計算託管遊戲或與 Amazon 配對的成本的資訊 GameLift，請參閱產生 Amazon GameLift 定價估算，其中說明如何使用 AWS Pricing Calculator.

亞馬遜如何GameLiftFlexMatch工作

本主題提供 Amazon GameLift FlexMatch 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 託GameLift管的遊戲，或搭配使用其他主機解決方案的遊戲使用。Amazon GameLift 伺服器上託管的遊戲 (包括即時伺服器) 會使用整合式 Amazon GameLift 服務來自動定位可用的遊戲伺服器，並開始比賽的遊戲工作階段。作FlexMatch為獨立服務使用的遊戲 (包括 Amazon GameLift FleetIQ) 必須與現有的託管系統協調，以指派託管資源並開始比賽的遊戲工作階段。

如需設定遊戲的FlexMatch詳細指引，請參閱FlexMatch 入門。

配對元件

FlexMatch配對系統包括以下部分或全部元件。

亞馬遜GameLift組件

這些是控制FlexMatch服務如何為您的遊戲執行配對的 Amazon GameLift 資源。它們是使用 Amazon GameLift 工具 (包括主控台和 AWS CLI) 建立和維護，或者也可以使用適用於 Amazon 的 AWS SDK 以程式設計方式進行建立和維護GameLift。

- FlexMatch配對配對配置（也稱為分房系統）— 分房系統是一組配置值，可為您的遊戲自訂配對程序。一個遊戲可以有多個配對系統，每個配對系統都根據需要配置為不同的遊戲模式或體驗。當您的遊戲傳送配對要求時FlexMatch，它會指定要使用哪個分房系統。
- FlexMatch配對規則集 — 規則集包含評估玩家是否有潛在比賽以及核准或拒絕所需的所有資訊。規則集定義了比賽的團隊結構、宣告用於評估的玩家屬性，並提供描述可接受比對標準的規則。規則可適用於個別玩家、隊伍或整場比賽。例如，一個規則可能要求比賽中的每個玩家都選擇相同的遊戲地圖，或者可能要求所有隊伍都具有相似的球員技能平均值。

- Amazon GameLift 遊戲工作階段佇列 (僅適FlexMatch用於 Amazon GameLift 受管主機) — 遊戲工作階段佇列會找到可用的託管資源，並為比賽啟動新的遊戲工作階段。佇列的組態決定 Amazon GameLift 尋找可用託管資源的位置，以及如何為比賽選擇最佳可用主機。

自訂元件

下列元件包含您必須根據遊戲架構實作的完整FlexMatch系統所需的功能。

- 配對的玩家介面 — 此介面可讓玩家加入比賽。它至少會透過用戶端配對服務元件啟動配對要求，並視配對程序所需提供玩家特定的資料，例如技能等級和延遲資料。

  > **Note**
  >
  > 最佳做法是，與FlexMatch服務的通訊應該由後端服務完成，而不是來自遊戲用戶端。

- 客戶端配對服務 — 此服務欄位玩家從玩家介面加入的請求，產生配對請求，並將其發送到服務。FlexMatch對於處理中的請求，它會監控配對事件，跟踪配對狀態，並根據需要採取行動。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。此元件會使用 AWS SDK 搭配 Amazon GameLift API 與FlexMatch服務進行通訊。
- 比賽放置服務（僅作FlexMatch為獨立服務）— 此組件與您現有的遊戲託管系統配合使用，以查找可用的託管資源並為比賽開始新的遊戲會話。元件必須取得配對結果，並擷取開始新遊戲工作階段所需的資訊，包括玩家 ID、屬性以及比賽中所有玩家的團隊指派。

FlexMatch配對過程

本主題說明基本的配對情境，以及各種遊戲元件與FlexMatch服務之間的互動。

要求玩家配對

使用您的遊戲客戶端的玩家點擊「加入遊戲」按鈕。此動作會導致您的用戶端配對服務傳送配對要求至。FlexMatch要求會識別完成要求FlexMatch時要使用的分房系統。請求也會包含您自訂分房系統所需的玩家資訊，例如技能等級、遊戲偏好設定或地理位置延遲資料。您可以為一位玩家或多位玩家提出配對要求。

將請求新增至配對池

當FlexMatch收到配對請求時，它會產生一張配對票券，並將其加入分房系統的票庫中。票證會保留在集區中，直到符合或達到最大時間限制為止。您的客戶配對服務會定期收到有關配對活動的通知，包括票證狀態的變更。

建立比賽

您的FlexMatch分房系統會持續針對其彩池中的所有票券執行下列程序：

1. 分房系統按票券年齡對池進行排序，然後開始從最古老的彩票開始建立一個潛在的比賽。
2. 分房系統會在潛在的對戰中新增第二張彩票，並根據您的自訂配對規則評估結果。如果潛在比賽通過評估，彩票的球員將被分配給一支球隊。
3. 分房系統會依序新增下一張票證，並重複評估程序。當所有玩家插槽都填滿後，比賽就準備就緒。

大型比賽（41 至 200 名玩家）的配對使用上述流程的修改版本，以便在合理的時間範圍內建立比賽。分房系統不會單獨評估每張彩票，而是將預先排序的彩票池分為潛在的比賽，然後根據您指定的玩家特徵來平衡每場比賽。例如，分房系統可能會根據類似的低延遲地點預先排序票券，然後使用比賽後平衡功能來確保隊伍依照玩家技能均勻配對。

報告配對結果

當找到可接受的匹配項時，所有匹配的票證都會更新，並為每個匹配的票證生成一個成功的配對活動。

- FlexMatch作為獨立服務：您的遊戲會在成功的配對活動中獲得比賽結果。結果數據包括所有匹配球員及其團隊分配的列表。如果您的比賽請求包含玩家延遲資訊，結果也會建議比賽的最佳地理位置。
- FlexMatch使用亞馬遜GameLift託管解決方案：比賽結果會自動傳遞到 Amazon GameLift 隊列以放置遊戲會話。分房系統會決定用於放置遊戲階段的佇列。

開始比賽的遊戲工作階段

建議的對戰成功形成後，就會開始新的遊戲工作階段。在為比賽設定遊戲工作階段時，您的遊戲伺服器必須能夠使用配對結果資料，包括玩家 ID 和團隊指派。

- FlexMatch作為獨立服務：您的自訂配對安置服務會從成功的配對活動中取得比賽結果資料，並連線至您現有的遊戲階段作業配置系統，找出可用的主機資源以供比賽使用。找到主機資源後，配對位置服務會與您現有的主機系統進行協調，以開始新的遊戲工作階段並取得連線資訊。
- FlexMatch使用亞馬遜GameLift託管解決方案：遊戲會話隊列可找到比賽的最佳可用遊戲服務器。視佇列的設定方式而定，它會嘗試將遊戲工作階段放置在成本最低的資源中，以及玩家會遇到低延遲的情況 (如果提供了玩家延遲資料)。成功放置遊戲工作階段後，Amazon GameLift 服務會提示遊戲伺服器開始新的遊戲工作階段，傳遞配對結果和其他可選的遊戲資料。

將玩家連接到比賽

在遊戲工作階段開始後，玩家會連線到工作階段、領取他們的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，為玩家提供連線資訊。
- FlexMatch使用 Amazon GameLift 託管解決方案：在成功放置遊戲工作階段時，使用遊戲工作階段連線資訊和玩家工作階段 ID FlexMatch 更新所有符合的票證。

FlexMatch 支援 AWS 區域

如果您 FlexMatch 與 Amazon GameLift 託管解決方案搭配使用，則可以在託管遊戲的任何位置託管相符的遊戲工作階段。查看 Amazon GameLift 託管的完整列表AWS 區域和位置。

對於所有 FlexMatch 使用者，您可以在下列支援AWS 區域的項目中託管 FlexMatch 資源，包括配對組態和規則集。請參閱 選擇分房系統的位置。

AWS 區域 name	區域代碼
美國東部 (維吉尼亞北部)	us-east-1
美國西部 (奧勒岡)	us-west-2
亞太區域 (首爾)	AP-东北 -2
亞太區域 (雪梨)	ap-southeast-2
亞太區域 (東京)	ap-northeast-1
歐洲 (法蘭克福)	eu-central-1
歐洲 (愛爾蘭)	eu-west-1
中國 (北京及寧夏地區)	

設定 FlexMatch

亞馬遜GameLiftFlexMatch是一種AWS服務，您必須擁有一個AWS帳戶才能使用此服務。建立 AWS 帳戶是免費的。如需有關您可以使用AWS帳戶執行哪些動作的詳細資訊，請參閱入門使用AWS。

如果您FlexMatch與其他 Amazon GameLift 解決方案搭配使用，請參閱下列主題：

- 設置亞馬遜GameLift託管和實時服務器的訪問權限
- 使用亞馬遜 GameLift FleetIQ 在亞馬遜 EC2 上設置託管的訪問權限

為亞馬遜設置您的帳戶 GameLift

1. 取得帳戶。打開亞馬遜網絡服務，然後選擇登錄到控制台。依照提示建立新帳戶或登入現有帳戶。
2. 設定管理使用者群組。開啟 AWS Identity and Access Management (IAM) 服務主控台，然後依照步驟建立或更新使用者或使用者群組。IAM 管理您的AWS服務和資源的存取權。使用 Amazon GameLift 主控台或呼叫 Amazon GameLift API 存取FlexMatch資源的每個人都必須獲得明確的存取權限。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指示，請參閱建立 IAM 使用者。
3. 將權限原則附加至您的使用者或使用者群組。透過將 IAM 政策附加到使用者或使用者群組來管理AWS服務和資源的存取。權限原則指定使用者必須存取的一組AWS服務和動作。

   對於 AmazonGameLift，您必須建立自訂許可政策，並將其附加到每個使用者或使用者群組。政策是 JSON 文件。使用以下範例建立您的政策。

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

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

FlexMatch 入門

使用本節中的資源來協助您開始使用建置配對系統。FlexMatch

主題

- 用於獨立配對的亞馬遜GameLiftFlexMatch集成
- FlexMatch與亞馬遜GameLift託管集成

用於獨立配對的亞馬遜GameLiftFlexMatch集成

本主題概述實作FlexMatch為獨立配對服務的完整整合程序。如果您的多人遊戲是使peer-to-peer用自訂設定的內部部署硬體或其他雲端運算原始型託管，請使用此程序。此程序也適用於 Amazon GameLift FleetIQ，這是亞馬遜 EC2 上託管遊戲的託管優化解決方案。如果您使用 Amazon 託管託GameLift管（包括即時伺服器）託管遊戲，請參閱FlexMatch與亞馬遜GameLift託管集成。

在開始整合之前，您必須擁有一個AWS帳戶並設定 Amazon GameLift 服務的存取權限。如需詳細資訊，請參閱 設定 FlexMatch。所有與建立和管理 Amazon GameLift FlexMatch 配對器和規則集相關的基本任務都可以使用 Amazon GameLift 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說明。在其中，您可以定義每個團隊的結構和大小。您也提供一組比賽必須符合才有效的要求，這些條件FlexMatch用於在比賽中包括或排除玩家。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，例如與最多 200 名玩家建立大型比賽。請參閱以下主題：

   - 建立FlexMatch規則集
   - FlexMatch 規則集範例

2. 設定配對活動的通知。使用通知追蹤FlexMatch配對活動，包括待處理的配對請求狀態。這是用來傳遞提議比對結果的機制。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首選選項。請參閱以下主題：

   - 設定FlexMatch事件通知
   - FlexMatch配對活動

3. 設置配對配FlexMatch對配置。也稱為分房系統，這個組件接收配對請求並處理它們。您可以透過指定規則集、通知目標和等待時間上限來設定分房系統。您也可以啟用選用功能。請參閱以下主題：

- 設計 FlexMatch 分房系統
- 建立配對規劃

4. 建立客戶端配對服務。建立或擴充遊戲用戶端服務，其中包含建立和傳送配對要FlexMatch求的功能。若要建立配對要求，此元件必須具備機制來取得配對規則集所需的玩家資料，以及選擇性地區延遲資訊。它還必須具有為每個請求創建和分配唯一票證 ID 的方法。您也可以選擇建立玩家接受工作流程，要求玩家選擇加入提議的比賽。此服務還必須監控配對活動，以獲得比賽結果，並啟動遊戲階段放置才能成功比賽。請參閱此主題：

   - 新增FlexMatch至遊戲用戶端

5. 建立配對安置服務。創建一個與您現有的遊戲託管系統配合使用的機制，以找到可用的託管資源並開始新的遊戲會話以成功比賽。此元件必須能夠使用比賽結果資訊來取得可用的遊戲伺服器，並為比賽開始新的遊戲工作階段。您可能還想實施一個工作流程來提出匹配的回填請求，該請求使用配對功能來填充已在運行的匹配遊戲工作階段中的空閒位置。

FlexMatch與亞馬遜GameLift託管集成

FlexMatch適用於自定義遊戲服務器和實時服務器的GameLift託管亞馬遜託管。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：
    - 建立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 分房系統

    - 建立配對規劃

- 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：
    - 新增FlexMatch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：
    - 添加FlexMatch到亞馬遜GameLift託管的遊戲服務器
- 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 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實時服務器的遊戲。

建立一個亞馬遜媒人 GameLift FlexMatch

FlexMatch 配對建構器程序會進行建置遊戲配對的工作。它可管理所收到配對建構請求的集區、組隊以進行配對、處理並選取玩家來尋找最佳可能玩家群組和為配對啟動遊戲工作階段預備與開始的程序。此主題描述創造配對建構器的主要層面以及如何為您的遊戲自訂一個配對建構器。

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配對過程。

主題

- 設計 FlexMatch 分房系統
- 建立FlexMatch規則集
- 建立配對規劃
- 設定FlexMatch事件通知

設計 FlexMatch 分房系統

本主題提供如何設計適合您遊戲的分房系統的指引。

設定基本分房系統

分房系統至少需要下列元素：

- 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 建立FlexMatch規則集 和 FlexMatch 規則集範例。
- 通知目標會收到所有配對活動通知。您需要設定 Amazon Simple Notification Service (SNS) 主題，然後將主題 ID 新增至分房系統。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 與 Amazon 託 GameLift 管搭配使用時，遊戲工作階段佇列會找到最佳可用資源來託管比賽的遊戲工作階段，並開始新的遊戲工作階段。每個佇列都會設定一份位置和資源類型清單 (包括 Spot 或隨需執行個體)，以決定可以放置遊戲工作階段的位置。如需佇列的詳細資訊，請參閱使用多位置佇列。

選擇分房系統的位置

決定您希望配對活動發生的位置，並在該位置建立配對規劃和規則集。Amazon 會針對您遊戲的比對請求 GameLift 維護票池，並針對可行的比賽進行排序和評估。進行比賽後，Amazon 會 GameLift 傳送遊戲工作階段配置的比賽詳細資訊。您可以在主機解決方案支援的任何位置執行相符的遊戲工作階段。

如FlexMatch 支援 AWS 區域需可建立 FlexMatch 資源的位置，請參閱。

在AWS 區域為您的分房系統選擇一個時，請考慮地點可能會如何影響表現，以及它如何為玩家最佳化比賽體驗。建議遵循下列最佳實務：

- 將分房系統放置在靠近您的玩家和客戶端服務的位置，以發送 FlexMatch 配對請求。這種方法可降低配對要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，請考慮在多個地點建立配對系統，並將配對請求傳送至離玩家最近的分房系統。除了提高效率之外，這會導致彩票池與地理位置靠近彼此的玩家形成，從而提高了分房系統根據延遲要求匹配玩家的能力。
- FlexMatch 與 Amazon 託 GameLift 管搭配使用時，請將您的分房系統和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。

添加可選元素

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 FlexMatch 用 Amazon GameLift 託管解決方案，則內置了許多功能。如果您使用 FlexMatch 的是獨立的配對服務，您可能會想要將這些功能建置到您的系統中。

玩家接受

您可以設定分房系統，要求所有被選中參加比賽的玩家都必須接受參與。如果您的系統需要接受，則必須給予所有玩家接受或拒絕提議的比賽的選項。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未能接受比賽，則提議的比賽將被丟棄，並按照以下方式處理門票。彩票中所有接受比賽的玩家將返回配對池以繼續處理的門票。至少有一位玩家拒絕比賽或無法回應的彩票將被置於失敗狀態且不再處理。接受玩家需要有時間限制；所有玩家必須在時限內接受建議的比賽，才能繼續比賽。

回填模式

使用 FlexMatch 回填功能讓您的遊戲工作階段在整個遊戲生命週期內充滿匹配良好的新玩家。處理回填請求時， FlexMatch 會使用與原始玩家相同的分房系統。您可以自定義如何使用新比賽的門票優先

排列回填門票，將回填票證放在該行的前端或末尾。這意味著，當新玩家進入配對池時，他們被放置在現有遊戲中的可能性或多或少於新組成的遊戲中。

無論您的遊戲與託管 Amazon GameLift 託管或其他託管解決方案 FlexMatch 搭配使用，都可以使用手動回填。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才新增玩家。

自動回填功能僅適用於使用 Amazon GameLift 託管託管的遊戲。啟用此功能後，如果遊戲工作階段以開放的玩家位置開始，Amazon 會開 GameLift 始自動產生回填請求。此功能可讓您設定配對機制，以便以最少的玩家數量開始新遊戲，然後在新玩家進入配對池時迅速填滿。您可以在遊戲工作階段期間隨時關閉自動回填功能。

遊戲屬性

對於 FlexMatch 搭配 Amazon GameLift 受管主機使用的遊戲，您可以在要求新的遊戲工作階段時提供其他資訊，以便傳送至遊戲伺服器。這是一個有用的方式來傳遞遊戲模式配置，這是為正在創建的比賽類型啟動遊戲會話所需的遊戲模式配置。由分房系統建立的所有對戰遊戲工作階段都會獲得相同的遊戲屬性。您可以通過創建不同的配對配置來更改遊戲屬性信息。

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。

建立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。一個團隊可以有新手和經驗豐富的球員，但兩隊的平均技能必須在 10 分以內彼此。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 主控台或 AWS CLI。

主題

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 名玩家的小型比賽建立規則集。配對規則集有兩件事：設定比賽的隊伍結構和規模，並告訴分房系統如何選擇玩家來組成最佳對戰。

但是你的配對規則集可以做得更多。例如，您可以：

- 為您的遊戲最佳化配對演算法。
- 設定最低玩家延遲要求，以保護遊戲品質。
- 隨著時間的推移逐漸放鬆團隊要求和比賽規則，以便所有活躍的玩家可以在需要時找到可接受的比賽。
- 使用對象彙總來定義群組配對要求的處理。
- 處理 40 個或更多玩家的大型比賽。如需建立大型相符項的更多資訊，請參閱〈〉設計大FlexMatch型比對規則集。

建立配對規則集時，請考慮下列選擇性和必要工作：

- 描述規則集 (必要)
- 自定義匹配算法
- 宣告玩家屬性
- 定義比賽團隊
- 設定玩家配對規則
- 允許要求隨著時間的推移放鬆

您可以使用 Amazon GameLift 主控台或`CreateMatchmakingRuleSet`作業建立規則集。

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。此值與您在使用 Amazon 建立規則集時指定的規則集名稱沒有關聯GameLift。
- ruleLanguageVersion— 用來建立FlexMatch規則的內容運算式語言版本。該值必須是`1.0`。

自定義匹配算法

FlexMatch最佳化大部分遊戲的預設演算法，讓玩家以最短的等待時間進入可接受的比賽。您可以自定義算法並調整遊戲的配對。

以下是預設的FlexMatch配對演算法：

1. FlexMatch將所有開放的配對票和回填票券放入票池中。
2. FlexMatch將彩池中的票券隨機分組成一個或多個批次。當票證集區變大時，會FlexMatch形成其他批次以維持最佳批次大小。
3. FlexMatch在每批中按年齡對門票進行排序。
4. FlexMatch根據每個批次的最舊票證建立匹配項。

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FlexMatch規則集:綱要的參考資訊，請參閱。

使用下列選用的自訂功能來影響配對程序的不同階段。

- 新增批次前排序
- 根據批次距離屬性的表單批次
- 優先順序回填工單
- 使用擴充功能支援較舊的門票

新增批次前排序

您可以在形成批次之前對票證集區進行排序。這種類型的自定義對於具有大型門票池的遊戲最有效。預批次分類可以幫助加快配對過程，並提高玩家在定義的特徵上的一致性。

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

自訂批次前排序的選項包括：

- 按播放器屬性排序。提供玩家屬性列表以預先排序彩票池。

若要依玩家屬性排序，batchingPreference請設定為sorted，並在中定義您的玩家屬性清單sortByAttributes。若要使用屬性，請先在規則集的playerAttributes元件中宣告屬性。

在下面的例子中，FlexMatch根據玩家偏好的遊戲地圖，然後按玩家技能對彩票池進行排序。產生的批次更有可能包含想要使用相同地圖的類似熟練玩家。

```
"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比對，或快速建立具有可接受延遲的比對。此自訂對於規則組成超過 40 名玩家的大型比賽很有用。

  將演算法屬性設strategy定為balanced。平衡策略會限制可用的規則陳述式類型。如需詳細資訊，請參閱設計大FlexMatch型比對規則集。

  FlexMatch使用下列其中一種方式，根據玩家報告的延遲資料來排序工單：

  - 最低延遲位置。彩票池是根據玩家報告其最低延遲值的地點進行預先排序。FlexMatch然後在相同位置以低延遲批次處理票證，創造更好的遊戲體驗。這也會減少每批次的票券數量，因此配對可能需要更長的時間。若要使用此自訂，請batchingPreference將設定為fastestRegion，如下列範例所示。

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

  - 可接受的延遲快速匹配。彩票池是按玩家報告可接受延遲值的地點進行預先排序。這會形成包含更多工單的批次較少。每批次中的票證越多，找到可接受的匹配會更快。若要使用此自訂，請batchingPreference將內容設定為 largestPopulation，如下列範例所示。

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

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

優先順序回填工單

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

回填優先級會影響它批次後如何FlexMatch處理票證。回填優先順序需要規則集使用詳盡搜尋策略。

FlexMatch不匹配多個回填票證在一起。

若要變更回填工單的優先順序，請設定屬性。`backfillPriority`

- 首先匹配回填門票。此選項嘗試在創建新匹配之前匹配回填票證。這意味著即將到來的玩家有更高的機會加入現有遊戲。

  如果您的遊戲使用自動回填功能，最好使用此功能。自動回填通常用於短遊戲會話和高玩家周轉的遊戲中。自動回填可以幫助這些遊戲形成最小的可行比賽，並在FlexMatch搜索更多玩家以填補空閒位置的同時讓他們開始。

  將 `backfillPriority` 設定為 `high`。

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

- 匹配回填門票最後。此選項會忽略回填工單，直到它評估所有其他工單為止。這意味著當新遊戲無法將新玩家匹配到現有遊戲中時，可以將其FlexMatch補充到現有遊戲中。

  當您想要使用回填作為最後機會的選項來吸引玩家進入遊戲時，這個選項非常有用，例如當玩家沒有足夠的玩家可以組成新的比賽時。

  將 `backfillPriority` 設定為 `low`。

  ```
  "algorithm": {
      "backfillPriority": "low",
  ```

```
    "strategy": "exhaustiveSearch"
},
```

使用擴充功能支援較舊的門票

當比賽難以完成時，擴展規則會放寬匹配條件。當部分完成的比賽中的門票達到特定年齡時，亞馬遜GameLift適用擴展規則。工單的創建時間戳確定 Amazon 何時GameLift應用規則；默認情況下，FlexMatch跟踪最近匹配的工單的時間戳。

若要變更FlexMatch套用展開規則的時間，請依下列方`expansionAgeSelection`式設定屬性：

- 根據最新票證進行擴展。此選項會根據新增至潛在比賽中的最新票證套用擴充規則。每次FlexMatch與新票證匹配時，時間時鐘都會被重置。使用此選項，產生的比對品質往往較高，但需要更長的時間才能符合；如果比對請求花費太長時間才能符合，則符合請求可能會在完成之前逾時。設定`expansionAgeSelection`為`newest`。 `newest`為預設值。
- 根據最舊的票證進行擴展。此選項會根據潛在比賽中最舊的票證套用擴充規則。使用此選項可以更快地FlexMatch套用資料片，從而縮短最早匹配玩家的等待時間，但會降低所有玩家的比賽品質。將 `expansionAgeSelection` 設定為 `oldest`。

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

宣告玩家屬性

在本節中，列出要包含在配對請求中的個別玩家屬性。您可能會在規則集中宣告玩家屬性的原因有兩個：

- 當規則集包含依賴玩家屬性的規則時。
- 當您想通過匹配請求將玩家屬性傳遞給遊戲會話時。例如，您可能希望在每個玩家連接之前將玩家角色選擇傳遞給遊戲會話。

宣告玩家屬性時，請包括下列資訊：

- name (必要) — 此值對於規則集而言必須是唯一的。
- type (必要) — 屬性值的資料類型。驗證資料類型為數字、字串或字串圖。

- default (選用) — 輸入配對請求未提供屬性值時要使用的預設值。如果沒有宣告預設值，且要求不包含值，則FlexMatch無法滿足要求。

定義比賽團隊

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 名玩家的潛在比賽是小型比賽，與 40 多名球員的比賽是大型比賽。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) — 為每個專案團隊指派一個唯一的名稱。您可以在規則和資料片中使用此名稱，以及遊戲工作階段中配對資料的FlexMatch參考資料。
- 最大玩家（必填）— 指定要分配給團隊的最大玩家數量。
- Min Player（必填）— 指定要分配給球隊的最小玩家數量。
- 數量 (選擇性) — 指定要使用此定義建立的專案團隊數目。當FlexMatch創建一個匹配，它給這些球隊提供的名稱和一個附加的數字。例如`Red-Team1`，`Red-Team2`、和`Red-Team3`。

FlexMatch嘗試將球隊填補到最大玩家規模，但確實創建球員較少的球隊。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`規則範例，請參閱FlexMatch 規則集範例主題。

設定玩家配對規則

建立一組規則陳述式，評估玩家是否接受比賽。規則可能會設定適用於個別玩家、隊伍或整個配對的要求。當 Amazon GameLift 處理比賽請求時，它會從可用玩家池中最古老的玩家開始，並圍繞該玩家建立比賽。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 (選擇性) — 使用此元素可附加自由格式文字描述。
- type (必要) — 類型元素識別處理規則時要使用的作業。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) — 根據定義的規則類型，您可能需要設定某些規則屬性。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。

允許要求隨著時間的推移放鬆

資料片FlexMatch可讓您在找不到相符項目的時間內放寬規則條件。此功能可確FlexMatch保在無法完美匹配時提供最佳功能。通過擴展放寬規則，您可以逐漸擴大可接受比賽的玩家池。

當未完成對戰中最新票券的年齡與擴充等待時間相符時，資料片就會開始。在比賽中FlexMatch添加新的票證時，擴展等待時間時鐘可能會被重置。您可以自訂規則集`algorithm`區段中展開的開始方式。

這是一個擴展的示例，它會逐漸提高比賽所需的最低技能等級。規則集使用距離規則語句，命名為SkillDelta要求一場比賽中的所有玩家都在彼此的 5 個技能等級以內。如果十五秒內沒有新的對戰，這個資料片會發現技能等級差異為 10，然後十秒後會尋找 20 的差異。

```
"expansions": [{
        "target": "rules[SkillDelta].maxDistance",
        "steps": [{
            "waitTimeSeconds": 15,
            "value": 10
        }, {
            "waitTimeSeconds": 25,
            "value": 20
        }]
    }]
```

啟用了自動回填功能的配對系統，不要太快地放鬆您的玩家數量需求。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之後開始擴展。擴展時間根據您的團隊組成而有所不同，因此請進行測試以找出適合您遊戲的最佳擴展策略。

設計大FlexMatch型比對規則集

如果您的規則集建立了允許 41 到 200 名玩家的匹配項，則需要對規則集配置進行一些調整。這些調整優化了匹配算法，以便它可以建立可行的大型比賽，同時還可以縮短玩家的等待時間。因此，大型相符規則集會使用針對一般配對優先順序進行最佳化的標準解決方案，取代耗時的自訂規則。

以下說明如何判斷您是否需要針對大型比對最佳化規則集：

1. 對於規則集中定義的每個團隊，獲取 MaxPlayer 的值，
2. 加起來所有的最大播放器值。如果總數超過 40，則表示您有很大的比對規則集。

若要針對大型比對最佳化規則集，請進行如下所述的調整。請參閱中的大型比對規則集的結構描述大型相符項目的規則集綱要和中的規則集範例範例 7：建立大型相符項目。

自定義大匹配的匹配算法

將演算法元件新增至規則集 (如果尚未存在)。設定下列屬性。

- strategy(必要) — 將內strategy容設定為「平衡」。此設定會觸發FlexMatch執行額外的比賽後檢查，以根據屬性中定義的指定玩家屬性來尋找最佳球隊平衡。balancedAttribute平衡策略取代了自定義規則以建立均勻匹配的團隊的需求。
- balancedAttribute（必要）— 確定在比賽中平衡球隊時要使用的球員屬性。此屬性必須具有數值資料類型 (雙精度或整數)。例如，如果您選擇在玩家技能上取得平衡，請FlexMatch嘗試指派玩家，以便所有隊伍都擁有盡可能均勻匹配的總技能等級。平衡屬性必須在規則集的播放器屬性中聲明。
- batchingPreference（可選）— 選擇您想要為玩家形成最低延遲比賽的重點。此設定會影響比賽票證在建立比賽項目之前的排序方式。選項包括：
  - 人口最多。FlexMatch允許使用池中至少一個共同位置具有可接受延遲值的所有票證進行比賽。因此，潛在的彩票池往往很大，這使得更容易更快地填寫比賽。玩家可能會以可接受的延遲（但並非總是最佳）進入遊戲中。如果未設置batchingPreference屬性，則設置為「平衡」時，這strategy是默認行為。
  - 最快的位置。FlexMatch根據報告最低延遲值的位置，預先排序集區中的所有工單。因此，比賽往往會由在相同位置回報低延遲的玩家組成。同時，每場比賽的潛在彩票池較小，這可能會增加填補比賽所需的時間。此外，由於延遲時間較高的優先順序，因此在平衡屬性方面，比賽中的玩家可能會有更大的差異。

下列範例會設定比對演算法，其行為如下：(1) 預先排序票證集區，依票證具有可接受延遲值的位置分組；(2) 形成批次排序的票證以進行比對；(3) 以批次建立票證比對，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

請確定您宣告在規則集演算法中做為平衡屬性的播放程式屬性。該屬性應該包含在配對請求中的每個玩家。您可以為播放程式屬性提供預設值，但是當提供玩家特定值時，屬性平衡效果最佳。

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。這會影響僅部分填滿時相符項目的外觀。您可能想要調整最小團隊規模以作回應。

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，即使比賽未滿，比賽中的球隊總是擁有接近相同數量的球員。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。在這種情況下，玩家最初被分配到擁有最多開放位置的最大球隊。隨著開放位置的數量變得更加均勻地分佈在所有團隊中，玩家將被插入較小的球隊中。

例如，假設您有一個包含三個團隊的規則集。紅色和藍色隊伍都設置為 `maxPlayers =10`，`minPlayers=5`。綠色小組被設置為 `maxPlayers =3`，`minPlayers=2`。以下是填充順序：

1. 尚未到達任何團隊`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)。現在兩支球隊都已經到達`minPlayers`。
2. 綠色團隊尚未到達`minPlayers`。接下來的 2 名玩家會指派給綠色隊伍。綠色團隊現在已經到達 `minPlayers`
3. 當所有隊伍都在時`minPlayers`，現在會根據開啟的欄位數量指派其他玩家。紅色和藍色隊伍各有 5 個開放位置，而綠隊則有 1 個。接下來的 8 名球員將被分配（每個 4 個）到紅隊和藍隊。所有隊伍現在都有 1 個開放欄位。
4. 剩餘的 3 個玩家插槽被分配（每個 1 個）以沒有特定的順序分配給團隊。

為大型比賽設定規則

大型比賽的配對主要依賴於平衡策略和延遲批次處理優化。因此大多數的自訂規則都無法使用。但是，您可以合併以下類型的規則：

- 對玩家延遲設定硬性限制的規則。將`latency`規則類型與屬性搭配使用`maxLatency`。請延遲規則參閱參考。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
```

```
    }],
```

- 根據指定播放器屬性中的親密關係來批處理玩家的規則。這與將平衡屬性定義為大型匹配算法的一部分不同，該算法著重於構建均勻匹配的團隊。此規則會根據指定屬性值 (例如初學者或專家技能) 中的相似性來批次配對票券，這往往會導致與指定屬性緊密對齊的玩家匹配。使用batchDistance規則類型，識別以數字為基礎的屬性，並指定允許的最大範圍。請批次距離規則參閱參考。以下是一個要求比賽的玩家在彼此的一個技能等級之內的範例：

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

放鬆大型比賽要求

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。在大型比賽中，您可以選擇放鬆延遲規則或團隊球員數量。

如果您在大型比賽中使用自動比賽回填功能，請避免過快地放鬆您的團隊球員數量。FlexMatch只有在遊戲工作階段開始後才開始產生回填要求，這可能會在比賽建立後數秒內不會發生。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
        }]
```

```
    }]
```

建立配對規則集

在為 Amazon 分房系統建立配對規則集之前，我們 GameLift FlexMatch 建議您先檢查規則集語法。使用 Amazon GameLift 主控台或 AWS Command Line Interface (AWS CLI) 建立規則集後，就無法變更它。

請注意，您可以在AWS區域中擁有的規則集數目上限有一個服務配額，因此最好刪除未使用的規則集。

相關主題

- 設計FlexMatch規則集
- FlexMatch 規則集範例
- FlexMatch規則語言

Console

建立規則集

1. 在以下位置打開 Amazon GameLift 控制台 https://console.aws.amazon.com/gamelift/。
2. 切換至您要在其中建立規則集的AWS區域。在與使用規則集的配對規則組態相同的區域中定義規則集。
3. 在導覽窗格中 FlexMatch，選擇「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，執行下列操作：
    a. 在「規則集設定」 下，針對「名稱」輸入唯一的描述性名稱，以便在清單或事件和量度表格中識別該名稱。
    b. 在規則集中，輸入 JSON 中的規則集。如需有關設計規則集的資訊，請參閱設計FlexMatch規則集。您也可以使用其中一個範例規則集FlexMatch 規則集範例。
    c. 選擇驗證以確認規則集的語法正確無誤。您無法在建立規則集之後編輯規則集，因此最好先驗證它們。
    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，您可以將規則集與分房系統一起使用。

AWS CLI

建立規則集

開啟指令行視窗並使用指令create-matchmaking-rule-set。

此範例指令會建立設定單一群組的簡單配對規則集。請務必在與使用該規則的配對規劃相同的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，Amazon 會 GameLift 傳回包含您指定之設定的MatchmakingRuleSet物件。分房系統現在可以使用新的規則集。

Console

刪除規則集

1. 在以下位置打開 Amazon GameLift 控制台 https://console.aws.amazon.com/gamelift/。
2. 切換至您在其中建立規則集的區域。
3. 在導覽窗格中 FlexMatch，選擇「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 會顯示錯誤訊息 (無法刪除規則集)。如果發生這種情況，請變更配對規劃以使用不同的規則集，然後再試一次。若要瞭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料頁面。

AWS CLI

刪除規則集

開啟指令行視窗，然後使delete-matchmaking-rule-set用指令刪除配對規則集。

如果配對組態正在使用規則集，Amazon 會 GameLift 傳回錯誤訊息。如果發生這種情況，請變更配對規劃以使用不同的規則集，然後再試一次。若要取得使用規則集的配對規則組態的清單，請使用指令describe-matchmaking-configurations並指定規則集名稱。

此範例指令會檢查配對規則集的用法，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場景。下列範例符合 FlexMatch 組態結構和屬性運算式語言。請根據需要完整複製這些規則集，或選擇其中的元件。

如需有關使用 FlexMatch 規則和規則集的詳細資訊，請參閱下列主題：

- 建立FlexMatch規則集
- 設計FlexMatch規則集
- FlexMatch規則集:綱要
- FlexMatch規則語言

> **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。

示例 1：創建兩個隊伍，並且匹配均勻的球員

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
        }]
    }]
}
```

示例 2：創建不均勻的團隊（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 3：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 暫定將潛在球員添加到團隊中，並計算團隊中球員的平均技能。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。
  - 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 4：使用明確排序來尋找最符合的項目

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。這些規則會對所有啟用中的配對票證進行排序，以根據特定的關鍵需求建立最佳配對。這個範例是透過下列指示來實作：

- 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 5：尋找涵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。示例遊戲是一種「free-for-all」風格，其中一場比賽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`運算來評估量測值中的每個字串清單如何與參照值中的字串清單相交。
- 測量 — 使用內`flatten`容運算式建立字串清單清單，每個字串清單都包含一個播放程式的 myCharacter 屬性值。
- 參照值 — 使用`set_intersection`內容表示式建立字串清單，其中包含相符項目中每個玩家通用的所有偏好設定屬性值。
- 限制 — 設定`minCount`為 1，以確保每位玩家選擇的角色（測量中的字串列表）至少與所有玩家共有的偏好對手中的一個匹配。（引用值中的字符串）。
- 擴充 — 如果在 15 秒內未填滿相符項，請放鬆最小交點需求。

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 會驗證測量值中的每個字串清單 (玩家選擇的字元) 與參考值 (玩家偏好的對手) 中的至少一個字串相交。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 6：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 7：建立大型相符項目

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 設計大FlexMatch型比對規則集 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 放鬆大型比賽要求 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 8：建立多隊大型比賽

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：

- 該規則集定義了可以容納多達 155 名玩家的團隊，這使其成為一場大型比賽。（10 x 15 獵人 + 5 個怪物 = 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。
- 以下是任何資料片生效前，此規則集的 FlexMatch 填滿符合方式：
  - 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 放鬆大型比賽要求 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 9：使用具有類似屬性的玩家建立大型對戰

此範例說明如何為使用兩支球隊的比賽設定規則集batchDistance。在範例中：

- 該SimilarLeague規則可確保一場比賽中的所有玩家都有 2 名其他玩家。league
- 該SimilarSkill規則可確保一場比賽中的所有玩家都有其他 skill 10 名玩家。如果玩家等待了 10 秒，則距離擴大到 20 秒。如果玩家等待了 20 秒，則距離會擴展到 40 秒。
- 該SameMap規則可確保一場比賽中的所有玩家都要求相同map。

- 該SameMode規則可確保一場比賽中的所有玩家都要求相同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 10：使用複合規則與具有類似屬性或類似選擇的玩家建立比賽

此範例說明如何為使用兩支球隊的比賽設定規則集compound。在範例中：

- 該SimilarLeagueDistance規則可確保一場比賽中的所有玩家都有 2 名其他玩家。league
- 該SimilarSkillDistance規則可確保一場比賽中的所有玩家都有其他 skill 10 名玩家。如果玩家等待了 10 秒，則距離擴大到 20 秒。如果玩家等待了 20 秒，則距離會擴展到 40 秒。
- 該SameMapComparison規則可確保一場比賽中的所有玩家都要求相同map。
- 該SameModeComparison規則可確保一場比賽中的所有玩家都要求相同mode。
- 如果至少符合下列條件之一，CompoundRuleMatchmaker則規則會確保相符：
  - 一場比賽中的玩家都要求相同map和相同的要求mode。
  - 比賽中的玩家具有可比league性skill和屬性。

```
{
    "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 11：建立使用玩家封鎖清單的規則

此範例說明了讓玩家避免與特定其他玩家配對的規則集。玩家可以建立一個封鎖清單，分房系統會在選擇一場比賽的玩家時評估該清單。如需有關新增封鎖清單或避免清單功能的詳細指引，請參[AWS閱遊戲部落格](#)。

此範例列出下列指示：

- 創建兩個正好由五名球員組成的團隊。
- 通過玩家的阻止列表，這是一個玩家 ID 列表（最多 100 個）。
- 將所有玩家與每個玩家的阻止名單進行比較，如果找到任何被阻止的玩家 ID，則拒絕提議的比賽。

使用此規則集的注意事項：

- 在評估新玩家以加入提議的比賽（或回填現有比賽中的位置）時，該玩家可能會因以下任一原因而被拒絕：
  - 如果新玩家在任何已被選中參加比賽的玩家的阻止列表中。
  - 如果任何已經被選中參加比賽的玩家在新玩家的阻止列表中。
- 如圖所示，此規則集可防止玩家與其封鎖清單中的任何玩家進行配對。您可以透過新增規則展開並增加`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
    }]
}
```

建立配對規劃

若要設定 Amazon GameLift FlexMatch 分房系統來處理配對請求，請建立配對組態。使用亞馬遜GameLift控制台或AWS Command Line Interface（AWS CLI）。如需建立分房系統的更多資訊，請參閱設計 FlexMatch 分房系統。

為亞馬遜GameLift託管創建媒人

在建立配對組態之前，請先建立規則集和 Amazon GameLift 遊戲工作階段佇列，以便與分房系統搭配使用。

Console

1. 在 Amazon 主GameLift控台的導覽窗格中，選擇配對組態。

2. 切換至您要建立分房系統的AWS區域。
3. 在配對規劃頁面上，選擇「建立配對規劃」。
4. 在「定義模型組態詳細資料」頁面的「配對規劃詳細資料」下，執行下列操作：

   a. 在名稱中，輸入可協助您在清單和指標中識別的分房系統名稱。分房系統名稱在區域中必須是唯一的。配對要求會依據其名稱和地區來識別要使用哪個分房系統。

   b. (選擇性) 在說明中，新增說明以協助識別分房系統。

   c. 在規則集中，從清單中選擇要與分房系統搭配使用的規則集。此清單包含您在目前「區域」中建立的所有規則集。

   d. 對於FlexMatch模式，請選擇「亞馬遜GameLift受管託管」。此模式會提示FlexMatch將成功的配對傳遞至指定的遊戲工作階段佇列。

   e. 在「AWS地區」中，選擇您設定要與分房系統搭配使用的遊戲工作階段佇列的地區。

   f. 在佇列中，選擇您要與分房系統搭配使用的遊戲工作階段佇列。

5. 選擇 下一步。
6. 在「規劃設定」 頁面的「配對設定」下，執行下列操作：

   a. 在「要求逾時」中，設定分房系統完成每個要求比對的時間上限 (以秒為單位)。FlexMatch取消超過此時間的配對要求。

   b. 對於回填模式，請選擇處理匹配回填的模式。

      - 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. (選擇性) 對於額外玩家數量，請設定比賽中要保持開啟的玩家欄位數量。FlexMatch將來可以用玩家填充這些插槽。

   d. （可選）在比賽接受選項下，對於「需要接受」，如果您想要求提議比賽中的每個玩家主動接受參與比賽，請選擇「必填」。如果您選擇此選項，則對於接受逾時，設定您希望分房系統在取消比賽前等待玩家接受的時間長度（以秒為單位）。

7. (選擇性) 在事件通知設定下，執行下列動作：

   a. (選擇性) 對於 SNS 主題，請選擇 Amazon 簡單通知服務 (Amazon SNS) 主題以接收配對事件通知。如果您尚未設定 SNS 主題，您可以稍後編輯配對設定來選擇。如需詳細資訊，請參閱設定FlexMatch事件通知。

   b. (選擇性) 對於自訂事件資料，請輸入您要在事件訊息中與此分房系統建立關聯的任何自訂資料。FlexMatch在與分房系統相關的每個活動中都會包含此資料。

8. (選擇性) 展開 [其他遊戲資料]，然後執行下列動作：

a. （選擇性）對於遊戲工作階段資料，請輸入任何您想要傳送給使用此配對設定FlexMatch進行的對戰開始的新遊戲工作階段的任何其他遊戲相關資訊。

b. (選擇性) 針對遊戲內容，新增包含新遊戲工作階段相關資訊的機碼值配對內容。

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 [檢閱並建立] 頁面上，檢閱您的選擇，然後選擇 [建立]。成功建立後，分房系統即可接受配對要求。

AWS CLI

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

這個範例指令會建立新的配對配對配置，需要玩家接受並啟用自動回填。它還保留了兩個玩家插槽供FlexMatch以後添加玩家，並提供了一些遊戲會話數據。

```
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"
```

如果配對組態建立請求成功，Amazon 會GameLift傳回一個MatchmakingConfiguration物件，其中包含您要求分房系統的設定。新的分房系統已準備好接受配對要求。

建立獨立的分房系統 FlexMatch

在建立配對組態之前，請先建立要與分房系統搭配使用的規則集。

Console

1. 在以下位置打開亞馬遜GameLift控制台 https://console.aws.amazon.com/gamelift/home。
2. 切換至您要建立分房系統的AWS區域。如需支援FlexMatch配對規劃的區域清單，請參閱選擇分房系統的位置。
3. 在導覽窗格中 FlexMatch，選擇配對組態。
4. 在配對規劃頁面上，選擇「建立配對規劃」。
5. 在「定義模型組態詳細資料」頁面的「配對規劃詳細資料」下，執行下列操作：

   a. 在名稱中，輸入可協助您在清單和指標中識別的分房系統名稱。分房系統名稱在區域中必須是唯一的。配對要求會依據其名稱和地區來識別要使用哪個分房系統。

   b. (選擇性) 在說明中，新增說明以協助識別分房系統。

   c. 在規則集中，從清單中選擇要與分房系統搭配使用的規則集。此清單包含您在目前「區域」中建立的所有規則集。

   d. 對於FlexMatch模式，請選擇「獨立」。這表示您具有自訂機制，可在 Amazon 以外的託管解決方案上啟動新遊戲工作階段GameLift。
6. 選擇 下一步。
7. 在「規劃設定」 頁面的「配對設定」下，執行下列操作：

   a. 在「要求逾時」中，設定分房系統完成每個要求比對的時間上限 (以秒為單位)。超過此時間的配對要求會遭到拒絕。

   b. （可選）在比賽接受選項下，對於「需要接受」，如果您想要求提議比賽中的每個玩家主動接受參與比賽，請選擇「必填」。如果您選擇此選項，則對於接受逾時，設定您希望分房系統在取消比賽前等待玩家接受的時間長度（以秒為單位）。
8. (選擇性) 在事件通知設定下，執行下列動作：

   a. (選擇性) 對於 SNS 主題，請選擇 Amazon SNS 主題以接收配對事件通知。如果您尚未設定 SNS 主題，您可以稍後編輯配對設定來選擇。如需詳細資訊，請參閱設定FlexMatch事件通知。

   b. (選擇性) 對於自訂事件資料，請輸入您要在事件訊息中與此分房系統建立關聯的任何自訂資料。FlexMatch在與分房系統相關的每個活動中都會包含此資料。
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 [檢閱並建立] 頁面上，檢閱您的選擇，然後選擇 [建立]。成功建立後，分房系統即可接受配對要求。

AWS CLI

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

此範例指令會為需要玩家接受的獨立分房系統建立新的配對配對配置。

```
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"
```

如果配對組態建立請求成功，Amazon 會GameLift傳回一個MatchmakingConfiguration物件，其中包含您要求分房系統的設定。新的分房系統已準備好接受配對要求。

編輯配對規劃

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

更新組態規則集時，如果存在現有的使用中配對工單，則新規則集可能不相容，原因如下：

- 新的或不同的團隊名稱或隊伍數量
- 新玩家屬性
- 變更現有玩家屬性類型

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規劃。

設定FlexMatch事件通知

您可以使用活動通知來追蹤個別配對要求的狀態。所有正在製作中或在製作中進行大量配對活動的遊戲都應使用事件通知。

設定事件通知有兩種選項。

- 讓您的分房系統將活動通知發佈至亞馬遜簡易通知服務 (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 發出的FlexMatch事件清單，請參閱FlexMatch配對活動。

設定EventBridge活動

亞馬遜GameLift會自動將所有配對活動發佈到亞馬遜EventBridge。使用EventBridge，您可以設定規則，將配對事件路由至目標進行處理。例如，您可以設定規則，將事件 "PotentialMatchCreated" 路由到處理玩家接受的AWS Lambda函數。如需詳細資訊，請參閱什麼是 AmazonEventBridge？

> **Note**
>
> 設定配對系統時，請將通知目標欄位保持空白，或者如果您要同EventBridge時使用 Amazon SNS，請參考 SNS 主題。

設置亞馬遜 SNS 主題

您可以讓亞馬遜將FlexMatch分房系統產生的所有事件GameLift發佈到 Amazon SNS 主題。

若要為亞馬遜GameLift事件通知建立 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，請參閱使用 Amazon 簡單通知服務開發人員指南AWS Management Console中的建立主題。
5. 在您主題的 [詳細資料] 頁面上，選擇 [編輯]。
6. (選擇性) 在您主題的 [編輯] 頁面上，展開 [存取政策]，然後將下列 AWS Identity and Access Management (IAM) 政策陳述式中的粗體語法新增至現有政策的結尾。（為了清晰起見，整個政策在此處顯示。）請務必針對您自己的 SNS 主題和 Amazon GameLift 配對組態使用 Amazon 資源名稱 (ARN) 詳細資料。

```
{
  "Version": "2008-10-17",
  "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:your_region:your_account:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:matchmakingconfiguration/
your_configuration_name"
        }
      }
    }
  ]
}
```

7. 選擇 Save changes (儲存變更)。

設定具有伺服器端加密的 SNS 主題

您可以使用伺服器端加密 (SSE) 在加密主題中儲存敏感資料。SSE 使用 AWS Key Management Service (AWS KMS) 中的受管金鑰來保護 Amazon SNS 主題中的訊息內容。如需使用 Amazon SNS 進行伺服器端加密的詳細資訊，請參閱 Amazon 簡單通知服務開發人員指南中的靜態加密。

若要設定具有伺服器端加密的 SNS 主題，請檢閱下列主題：

- 在AWS Key Management Service開發人員指南中建立金鑰
- 在 Amazon 簡易通知服務開發人員指南中的主題啟用 SSE

建立 KMS 金鑰時，請使用下列 KMS 金鑰原則：

```
{
  "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

您可以使用發佈到 Amazon SNS 主題的事件通知來叫用 Lambda 函數。設定分房系統時，請務必將通知目標設定為 SNS 主題的 ARN。

下列AWS CloudFormation範本會將訂閱設定為名為的 SNS 主題，以叫用名`MyFlexMatchEventTopic`為的 Lambda 函數。`FlexMatchEventHandlerLambdaFunction`範本會建立 IAM 許可政策，GameLift讓亞馬遜寫入 SNS 主題。然後，範本會新增 SNS 主題的權限，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
```

準備遊戲 FlexMatch

使用 Amazon GameLift FlexMatch 將玩家配對功能新增至您的遊戲。FlexMatch可與自定義遊戲服務器和實時服務器的託管亞馬遜GameLift解決方案一起使用。

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。

FlexMatch 以佇列功能為建構基礎。配對形成後，FlexMatch 就會將配對詳細資訊交給您選擇的佇列。佇列會搜尋 Amazon GameLift 叢集上可用的託管資源，並為比賽開始新的遊戲工作階段。

本節的主題會說明如何對遊戲伺服器和遊戲用戶端新增配對支援。若要為遊戲建立配對建構器，請參閱建立一個亞馬遜媒人 GameLift FlexMatch。如何 FlexMatch 運作方式的詳細資訊，請參閱 亞馬遜如何GameLiftFlexMatch工作。

新增FlexMatch至遊戲用戶端

本主題說明如何將FlexMatch配對支援新增至您的用戶端遊戲服務。無論您是FlexMatch搭配 Amazon託管主機還是搭配其他託GameLift管解決方案使用，程序基本上都是相同的。若要進一步了解 FlexMatch 與如何為遊戲設定自訂配對建置器，請參閱下列主題：

- FlexMatch與亞馬遜GameLift託管集成
- 亞馬遜如何GameLiftFlexMatch工作
- 建立一個亞馬遜媒人 GameLift FlexMatch
- FlexMatch 規則集範例

若要在您的遊戲中啟用FlexMatch配對功能，請新增下列功能：

- 準備要求一個或多個玩家配對（必填）。
- 追蹤配對要求的狀態 (必填)。
- 要求玩家接受提議的比賽（可選）。
- 在為新比賽建立遊戲工作階段後，取得玩家連線資訊並加入遊戲。

準備為玩家申請配對

我們強烈建議您的遊戲用戶端透過用戶端遊戲服務提出配對要求。藉由使用受信任的來源，您可以更輕鬆地防止駭客嘗試入侵和仿造玩家的資料。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。

為了準備您的用戶端服務，請執行下列工作：

- 添加亞馬遜 GameLift API。您的用戶端服務會使用 Amazon GameLift API 中的功能，這是AWS開發套件的一部分。如需進一步了解開發AWS套件並下載最新版本，請參閱 Amazon GameLift SDK 以取得用戶端服務的開發套件。將此開發套件新增到您的遊戲用戶端服務專案。
- 設置配對票系統。會對所有配對請求指派一個專屬的票證 ID。您需要可用來產生唯一 ID，並將其指派給新配對請求的機制。票證 ID 可使用任何字串格式，最長可達 128 個字元。
- 取得遊戲建置器資訊。取得您計畫使用的配對組態名稱。您也需要配對建構器的必要玩家屬性清單，而這些屬性會定義於配對建構器的規則集中。
- 取得玩家資料。設定取得每個玩家相關資料的方法。這包括玩家 ID、玩家屬性值，以及玩家可能進入遊戲的每個區域更新延遲資料。
- (選用) 啟用配對回填。決定您想要回填現有配對遊戲的程度。如果您的配對建置器已經將回填模式設為「手動」(manual)，您可能需要將回填支援新增到您的遊戲。如果回填模式設為「自動」(automatic)，您可能需要針對個別遊戲工作階段關閉該模式的方法。進一步了解管理配對回填：使用回填現有遊戲 FlexMatch。

要求玩家配對

將程式碼新增到您的用戶端服務，以便為 FlexMatch 服務建置器建立和管理配對請求。對於搭FlexMatch配 Amazon GameLiftmanaged 主機使用的遊戲以及作FlexMatch為獨立解決方案使FlexMatch用的遊戲，請求配對的程序是相同的。

建立配對請求：

- 調用亞馬遜 GameLift API StartMatchmaking。每個請求都必須包含下列資訊。

  配對建構器

  要用於請求的配對配對組態名稱。FlexMatch將每個請求放入指定分房系統的集區中，然後根據配置分房系統的方式處理要求。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 分房系統。

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 — 每個玩家都必須有一個唯一的玩家 ID，您可以產生該 ID。請參閱產生玩家 ID。
- 玩家屬性 — 如果使用中的分房系統要求玩家屬性，請求必須為每位玩家提供這些屬性。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。在建立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 — 如果使用中的分房系統有玩家延遲規則，請求必須回報每位玩家的延遲時間。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。

擷取配對請求詳細資訊：

- 發送匹配請求後，您可以通過調DescribeMatchmaking用請求的工單 ID 來查看請求詳細信息。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。

取消配對請求：

- 您可以隨時透過撥打StopMatchmaking要求的票證 ID 來取消配對要求。

追蹤配對活動

設定通知以追蹤 Amazon 針對配對程序GameLift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DescribeMatchmaking用要求的票證 ID 撥打電話，擷取配對請求票證 (包括目前狀態)。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 代管主機使用的遊戲以及FlexMatch作為獨立解決方案使用的遊戲，管理玩家接受程序是相同的。

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。對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。您可以通過調用來檢索匹配詳細信息DescribeMatchmaking。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。通過撥打AcceptMatch接受或拒絕來報告玩家回應。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。確實接受比賽的玩家的門票將自動返回到彩票池中。不接受比賽的玩家的門票將移至「失敗」狀態且不再處理。對於有多名玩家的彩票，如果彩票中有任何玩家不接受比賽，則整張彩票將失敗。

連接到匹配

將程式碼新增至您的用戶端服務，以處理成功格式的相符項目 (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 託GameLift管的遊戲，當配對請求成功完成時，遊戲工作階段連線資訊會新增至配對票證中。透過呼叫DescribeMatchmaking擷取已完成的配對票券。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。您的遊戲客戶端可以使用此信息直接連接到比賽的遊戲會話。連線要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比賽資料相關聯，其中包括團隊指派 (請參閱 GameSession)。

對於使用其他託管解決方案的遊戲 (包括 Amazon GameLift FleetIQ)，您必須建置機制，讓配對玩家能夠連線到適當的遊戲工作階段。

配對要求範例

下列程式碼片段會針對數個不同的配對系統建立配對要求。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)
```

添加FlexMatch到亞馬遜GameLift託管的遊戲服務器

本主題說明如何為使用 Amazon 託GameLift管的自訂遊戲伺服器新增FlexMatch配對支援。若要進一步了解將 FlexMatch 新增到遊戲的資訊，請參閱下列主題：

- 亞馬遜如何GameLiftFlexMatch工作
- FlexMatch與亞馬遜GameLift託管集成

本主題中的資訊假設您已成功將 Amazon GameLift Server SDK 整合到遊戲伺服器專案中，如將 Amazon 新增GameLift至遊戲伺服器中所述。此項工作完成時，您將擁有大部分所需的機制。本主題的區段涵蓋剩餘的工作，以便處理使用 FlexMatch 設定的遊戲。

設定您的遊戲伺服器以進行配對

設定遊戲伺服器處理配對遊戲時，請完成下列作業。

1. 啟動使用配對所建立的遊戲工作階段。若要請求新的遊戲工作階段，Amazon GameLift 會透過遊戲工作階段物件向您的遊戲伺服器傳送`onStartGameSession()`請求 (請參閱 GameSession)。遊戲伺服器使用遊戲工作階段資訊，包括自訂遊戲資料在內，以啟動要求的遊戲工作階段。如需詳細資訊，請參閱「開始遊戲工作階段」。

   至於在配對遊戲方面，遊戲工作階段物件也包含一組配對建構器資料。配對建構器資料之中包含資訊，可讓遊戲伺服器在開始新遊戲工作階段時進行配對。其中包括配對的團隊結構、團隊指派及可能與您遊戲相關的特定玩家屬性。例如，您的遊戲可能依據平均玩家技能層級解鎖特定功能或層級，或根據玩家的喜好設定選擇一張地圖。請至 使用分房系統資料 進一步了解。
2. 處理玩家連線。連線到符合的遊戲時，遊戲用戶端會引用玩家 ID 和玩家工作階段 ID（請參閱驗證新玩家）。您的遊戲伺服器會使用玩家 ID，以便將傳入的玩家與配對建構器資料中的玩家資訊產生關連。配對建構器資料識別玩家的團隊指派，並可能提供其他資訊，以便正確地在遊戲中代表玩家。
3. 玩家離開遊戲時進行回報。確認您的遊戲伺服器正在呼叫伺服器 API `RemovePlayerSession()`來回報掉落的玩家 (請參閱回報玩家工作階段結束)。如果您在現有遊戲中使用 FlexMatch 回填來填入空插槽，此步驟便很重要。如果您的遊戲透過用戶端遊戲服務啟動回填請求，則此步驟至為關鍵。如需實作 FlexMatch 回填功能的詳細資訊，請參閱使用回填現有遊戲 FlexMatch。
4. 為現有的配對遊戲工作階段請求新玩家 (選用)。決定您想要回填現有配對遊戲的程度。如果您的配對建置器已經將回填模式設為「手動」(manual)，您可能需要將回填支援新增到您的遊戲。如果回

填模式設為「自動」(automatic)，您可能需要針對個別遊戲工作階段關閉該模式的方法。例如，您可能希望在遊戲達到特定時間點之後，就停止回填遊戲工作階段。進一步了解管理配對回填：使用回填現有遊戲 FlexMatch。

使用分房系統資料

您的遊戲伺服器必須能夠辨識並使用GameSession物件中的遊戲資訊。只要遊戲工作階段啟動或更新，Amazon GameLift 服務就會將這些物件傳遞至您的遊戲伺服器。核心遊戲工作階段資訊包括遊戲工作階段 ID 及名稱、最大玩家數量、連線資訊及自訂遊戲資料 (若提供)。

對於使用 FlexMatch 建立的遊戲工作階段，GameSession 物件還會包含一組配對建構器資料。除了唯一的配對 ID，資料也會識別配對建構器，其會建立配對，並說明團隊、團隊指派及玩家。其中包括原始配對要求的玩家屬性 (請參閱 Player (玩家) 物件)。其中並未包括玩家延遲；若您需要現有玩家的延遲資料 (例如用於配對回填)，我們建議您取得新資料。

> **Note**
>
> 分房系統資料會指定完整的配對配置 ARN，可識別配置名稱、AWS帳號和地區。從遊戲用戶端或服務請求配對回填時，則僅需要組態名稱。您可擷取組態名稱，方法為剖析「:matchmakingconfiguration/」之後的字串。在所示範例中，配對規劃名稱為「MyMatchmakerConfig」。

下列 JSON 顯示一般的配對資料集。本範例說明兩位玩家的遊戲，玩家配對是以技術評等和獲得的最高等級為依據。配對建構器也依據角色配對，並確保配對玩家至少有一個共同的對應偏好設定。在這種情況下，遊戲伺服器應該能夠判定哪些地圖是最慣用的地圖並用於遊戲工作階段。

```
{
 "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}}
   }
  }]
 }]
}
```

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中（請參閱「讓玩家加入遊戲階段」），但比賽回填功能可確保新玩家符合與目前玩家相同的比賽條件。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配對過程。

> **Note**
>
> FlexMatch回填目前不適用於使用實時服務器的遊戲。

回填機制有兩種類型：

- 若要填滿以少於允許玩家上限開始的遊戲工作階段，請啟用自動回填。
- 若要取代退出進行中遊戲工作階段的玩家，請在遊戲伺服器中新增功能以傳送回填請求。

開啟自動回填

透過自動比賽回填功能，只要遊戲工作階段以一或多個未填滿的玩家插槽開始，Amazon GameLift 就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。

舉例來說，請考量一個可容納六到十名玩家的遊戲。FlexMatch最初定位六名玩家，組成比賽，並開始一個新的遊戲會話。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

格的不同，我們可能希望在遊戲階段期間允許新玩家隨時加入。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，它會用於使用該分房系統建立的所有相符遊戲工作階段。只要遊戲工作階段在遊戲伺服器上啟動，Amazon 就會GameLift開始為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「建立配對規劃」。
2. 開啟回填優先順序。自訂您的配對流程，以便在建立新相符項目之前優先填寫回填請求。在配對規則集中，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 自定義匹配算法。
3. 使用新的分房系統資料更新遊戲工作階段。亞馬遜使用 Server SDK 回調函數使用匹配信息GameLift更新您的遊戲服務器onUpdateGameSession（請參閱初始化服務器過程）。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比賽資料 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，請將程式碼新增至遊戲用戶端或遊戲伺服器以進行 Amazon GameLift 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。

發送回填請求（從遊戲服務器）

您可以直接從代管遊戲工作階段的遊戲伺服器程序來初始化配對回填請求。服務器進程包含有up-to-date關當前玩家連接到遊戲的最多信息以及空白玩家插槽的狀態。

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 FlexMatch與亞馬遜GameLift託管集成。

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比賽資料。

與其他伺服器功能一樣，遊戲伺服器會使用 Amazon GameLift 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（或選擇讓它們自動生成）。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。
   - 分房系統 — 識別要使用哪個分房系統進行回填請求。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。此信息存儲在遊戲會話對象（GameSession）中，該對象在激活遊戲會話GameLift時由亞馬遜提供給服務器進程。配對建構組態 ARN 包含在 `MatchmakerData` 屬性中。
   - 遊戲工作階段 ARN — 識別正在回填的遊戲工作階段。您可以通過調用服務器 API GetGameSessionId（）來獲取遊戲會話 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。
   - 玩家數據 -包括您正在回填的遊戲會話中所有當前玩家的玩家信息（玩家）。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。您必須包括每位玩家的團隊成員資格。如果您不使用回填，請勿指定群組。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. 主控遊戲工作階段的伺服器程序應該擁有最多玩家目前連線到遊戲工作階段的up-to-date資訊。
     2. 要獲取玩家 ID，屬性和團隊分配，請從遊戲會話對象（GameSession）`MatchmakerData`屬性中提取玩家數據（請參閱使用分房系統資料）。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. 追蹤回填要求的狀態。亞馬遜使用 Server SDK 回調函數GameLift更新您的遊戲服務器有關回填請求的狀態onUpdateGameSession（請參閱初始化服務器進程）。新增程式碼以處理狀態訊息，以及因成功回填要求而更新的遊戲工作階段物件，位於。更新遊戲伺服器上的比賽資料

   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，請致電 StopMatchBackfill()。如果您需要變更請求，請呼叫 StopMatchBackfill 接著提交更新請求。

發送回填請求（從客戶端服務）

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 FlexMatch與亞馬遜GameLift託管集成。

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比賽資料

與其他用戶端功能一樣，用戶端遊戲服務會使用 AWS SDK 搭配 Amazon GameLift API。SDK 有 C++、C# 及多種其他語言的選擇。如需用戶端 API 的一般說明，請參閱 Amazon GameLift 服務 API 參考，其中說明適用於 Amazon GameLift 相關動作的低階服務 API，並包含特定語言參考指南的連結。

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。您可以通過DescribeGameSessions使用遊戲會話 ID 調用從遊戲會話對象（GameSession）獲取此信息。

- 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。每當您的遊戲伺服器聯絡 Amazon GameLift 服務以驗證新玩家連線或回報玩家掉線時，目前的玩家人數就會更新。

- 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 要求玩家配對)，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（或選擇讓它們自動生成）。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。
   - 分房機制 — 識別要使用的配對配對配置名稱。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。這些信息是在一個遊戲會話對象（GameSession），`MatchmakerData`屬性下的配對配置 ARN。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。回填要求的配對票券 (MatchmakingTicket) 會在處理時具有遊戲工作階段 ID；新配對要求的票券在進行比賽前不會取得遊戲工作階段 ID；在遊戲工作階段 ID 的出現是分辨新比賽門票與補充票券之間差異的一種方式。
   - 玩家數據 -包括您正在回填的遊戲會話中所有當前玩家的玩家信息（玩家）。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。您必須包括每位玩家的團隊成員資

格。如果您不使用回填，請勿指定群組。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. 使用遊戲工作階段 ID 呼叫 DescribePlayerSessions()，以發現目前連線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. 從遊戲會話對象中提取玩家數據（GameSession），MatchmakerData屬性（見使用分房系統資料。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 追蹤配對活動)。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比賽資料 了解更多資訊。

更新遊戲伺服器上的比賽資料

無論您如何在遊戲中發起匹配回填請求，您的遊戲伺服器都必須能夠處理 Amazon 因匹配回填請求而 GameLift提供的遊戲工作階段更新。

當 Amazon GameLift 完成匹配回填請求（無論是否成功），它都會使用回調函數調用您的遊戲服務器。onUpdateGameSession此呼叫有三個輸入參數：比賽回填票證 ID、狀態訊息，以及包含最多 up-to-date配GameSession對資料 (包括玩家資訊) 的物件。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)：
   - 配對資料更新 — 新玩家已成功配對遊戲階段。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)。

- 回填失敗 — 比對回填嘗試失敗，因為內部錯誤。GameSession 物件保持不變。
- 回填 _ TIMED_OUT — 分房系統無法在時間限制內找到回填相符項目。GameSession 物件保持不變。
- 回填 _ 取消 — 對 StopMatchmaking (用戶端) 或 StopMatchBackfill (伺服器) 的呼叫已取消比對回填要求。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. 在您的遊戲伺服器呼叫 Server SDK 動作 ProcessReady() 中，新增onUpdateGameSession回呼方法名稱做為處理序參數。

亞馬遜GameLiftFlexMatch參考

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

主題

- 亞馬遜 GameLift FlexMatch API 參考 (AWS開發套件)
- FlexMatch規則語言
- FlexMatch配對活動

亞馬遜 GameLift FlexMatch API 參考 (AWS開發套件)

本主題為 Amazon 提供以工作為基礎的 API 操作清單。GameLift FlexMatch亞馬遜GameLiftFlexMatch服務 API 封裝到`aws.gamelift`命名空間中的AWS開發套件中。下載開AWS發套件或檢視亞馬遜 GameLift API 參考文件。

Amazon GameLift FlexMatch 提供配對服務，可與 Amazon 託管解決方案GameLift託管的遊戲搭配使用 (包括自訂遊戲伺服器或即時伺服器的受管託管，以及使用 Amazon GameLift FleetIQ 在 Amazon EC2 上託管)，以及與其他託管系統 (例如peer-to-peer現場部署或雲端運算原語) 一起使用。如需其他 Amazon GameLift 託管選項的詳細資訊，請參閱 Amazon GameLift 開發人員指南。

設定配對規則和流程

調用這些行動來建立FlexMatch分房系統、設定遊戲的配對程序，以及定義一組自訂規則來建立比賽和團隊。

配對組態

- CreateMatchmakingConfiguration— 創建一個配對配置，其中包含評估玩家組和建立球員團隊的說明。使用 Amazon 託管GameLift時，還要指定如何為比賽建立新的遊戲工作階段。
- DescribeMatchmakingConfigurations— 擷取定義了 Amazon GameLift 區域的配對組態。
- UpdateMatchmakingConfiguration— 更改配對配置的設置。隊列。
- DeleteMatchmakingConfiguration— 從區域移除配對配對配置。

配對規則集

- CreateMatchmakingRuleSet— 創建一組搜索球員匹配時使用的規則。

- DescribeMatchmakingRuleSets— 擷取 Amazon GameLift 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。

為一名或多位玩家申請比賽

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— 請求一名玩家或想要在同一場比賽中進行比賽的小組配對。
- DescribeMatchmaking— 獲取有關配對請求的詳細信息，包括狀態。
- AcceptMatch— 對於需要玩家接受的比賽，請在玩家接受建議的比賽GameLift時通知 Amazon。
- StopMatchmaking— 取消配對要求。
- StartMatchBackfill-請求其他玩家匹配以填充現有遊戲會話中的空白位置。

可用的程式設計語言

支援亞馬遜的AWS開發套件GameLift提供下列語言版本。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) (亞馬遜 GameLift)
- Java（SDK 文檔）（亞馬遜 GameLift）
- .NET（SDK 文檔）（亞馬遜 GameLift）
- 去（SDK 文檔）（亞馬遜 GameLift）
- 蟒蛇 (SDK 文檔) (亞馬遜 GameLift)
- 紅寶石（SDK 文檔）（亞馬遜 GameLift）
- PHP（SDK 文檔）（亞馬遜 GameLift）
- JavaScript/Node.js (開發套件文件) (亞馬遜 GameLift)

FlexMatch規則語言

本節中的參考主題說明用於建立配對規則以與 Amazon 搭配使用的語法和語意。GameLift FlexMatch 如需撰寫配對規則和規則集的詳細說明，請參閱建立FlexMatch規則集。

主題

FlexMatch規則集:綱要

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描述，請參閱FlexMatch規則集性質定義。

小型相符項目的規則集綱要

以下結構描述記錄規則集的所有可能屬性和允許的值，這些規則集用於建立最多 40 位玩家的相符項目。

```
{
    "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
        }]
    }]
}
```

大型相符項目的規則集綱要

下列結構描述記錄規則集的所有可能屬性和允許的值，這些規則集用來建立超過 40 位玩家的相符項目。如果規則集中所有專案團隊的總maxPlayers值超過 40，則FlexMatch程序會比對使用大型比對準則下使用此規則集的要求。

```
{
    "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規則集性質定義

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，請參閱建立FlexMatch規則集。

name

規則集的描述性標籤。此值與指派給 Amazon GameLift MatchmakingRuleSet資源的名稱無關聯。此值包含在描述完成比對的配對資料中，但不會被任何 Amazon GameLift 程序使用。

允許的值：字串

是否為必要？ 否

ruleLanguageVersion

所使用的FlexMatch屬性表示式語言版本。

允許的值：「1.0」

是否為必要？ 是

playerAttributes

包含在配對要求中的玩家資料集合，並在配對過程中使用。您也可以在此處宣告屬性，讓玩家資料包含在傳送至遊戲伺服器的配對資料中，即使資料並未在配對過程中使用。

是否為必要？ 否

name

分房系統使用的玩家屬性的唯一名稱。此名稱必須與配對要求中參照的玩家屬性名稱相符。

允許的值：字串

是否為必要？ 是

type

播放器屬性值的數據類型。

允許的值：「字符串」，「數字」，「字符串列表」，「字符串數字 _ 映射」

是否為必要？ 是

default

配對請求未提供給玩家時所使用的預設值。

允許的值：播放程式屬性允許的任何值。

是否為必要？ 否

algorithm

用於自訂配對程序的選擇性組態設定。

是否為必要？ 否

strategy

建立相符項目時要使用的方法。如果未設定此屬性，預設行為是「詳盡搜尋」。

允許的值：

- 「詳盡搜索」— 標準匹配方法。FlexMatch透過根據一組自訂比對規則評估池中的其他工單，圍繞批次中最舊的工單組成比對。此策略用於 40 名以下玩家的比賽。使用此策略時，batchingPreference應設置為「隨機」或「排序」。
- 「平衡」— 優化以快速形成大型比賽的方法。此策略僅用於 41 至 200 名玩家的比賽。它通過預先排序彩票池，建立潛在的比賽並將球員分配給團隊，然後使用指定的球員屬性在比賽中平衡每個球隊來形成比賽。例如，此策略可用於平衡一場比賽中所有隊伍的平均技能水平。使用此策略時，balancedAttribute必須設置，並且batchingPreference應該設置為「最大人口」或「最大人口」。大多數自訂規則類型無法透過此策略辨識。

是否為必要？ 是

batchingPreference

在將票證分組以進行比賽建立之前使用的預先排序方法。預先排序彩票池會導致彩票根據特定特徵進行批次處理，這往往會增加最終比賽中各玩家的一致性。

允許的值：

- 「隨機」-僅在 strategy =「詳盡搜索」時有效。沒有預先排序; 池中的門票是隨機批量的。這是詳盡搜尋策略的預設行為。
- 「排序」-僅適用於 strategy =「詳盡搜索」。彩票池根據中sortbyAttributes列出的玩家屬性進行預先排序。
- 「人口最大」-僅在 strategy =「平衡」的情況下有效。彩票集區會依玩家報告可接受延遲等級的地區進行預先排序。這是平衡策略的預設行為。
- 「快速傳統」— 僅在 strategy =「平衡」的情況下有效。彩票集區會依玩家報告其最低延遲等級的地區進行預先排序。由此產生的比賽需要更長的時間才能完成，但所有玩家的延遲時間往往很低。

是否為必要？ 是

balancedAttribute

使用平衡策略建立大型比賽時要使用的玩家屬性名稱。

允許的值：在中宣告playerAttributes的任何屬性 type =「數字」。

是否為必要？ 是的，如果 strategy =「平衡」。

sortByAttributes

在批次處理之前預先排序票證池時要使用的玩家屬性列表。只有在使用詳盡搜尋策略進行預先排序時，才會使用此屬性。屬性清單的順序決定了排序順序。FlexMatch使用字母和數值的標準排序慣例。

允許的值：在中宣告的任何屬性playerAttributes。

是否為必要？ 是的，如果 batchingPreference =「排序」。

backfillPriority

比對回填工單的優先順序排列方法。此屬性決定何時FlexMatch處理批次回填工單。僅在使用詳盡搜索策略進行預排序時使用。如果未設定此屬性，則預設行為為為「normal」。

允許的值：

- 「一般」— 在組成比賽時，不會考慮票券的請求類型（回填或新配對）。
- 「高」— 工單批次會依請求類型 (然後按年齡) 排序，並FlexMatch嘗試先比對回填票證。
- 「low」— 工單批次會依要求類型 (然後按年齡排序) 排序，並FlexMatch嘗試先比對非回填票證。

是否為必要？ 否

expansionAgeSelection

用於計算比對規則擴充等待時間的方法。如果在一段時間過後仍未完成比賽，資料片將用於放鬆對戰需求。等待時間是根據已在部分填寫的比賽中的門票年齡計算的。如果未設定此屬性，預設行為是「最新」。

允許的值：

- 「最新」— 擴充等待時間是根據在部分完成比賽中具有最新建立時間戳記的工單來計算。擴充的觸發速度往往較慢，因為一個較新的工單可以重新啟動等待時間時鐘。
- 「最舊」— 擴充等待時間是根據比賽中建立時間戳記最早的票證來計算。資料片往往會更快地觸發。

是否為必要？ 否

teams

比賽中團隊的配置。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群組。

name

專案團隊的唯一名稱。專案團隊名稱可以在規則和展開中參照。在成功的對戰中，玩家會在配對資料中依隊伍名稱指派。

允許的值：字串

是否為必要？ 是

maxPlayers

可以分配給團隊的玩家數量上限。

允許的值：數字

是否為必要？ 是

minPlayers

比賽開始前必須分配給球隊的最低球員數量。

允許的值：數字

是否為必要？ 是

quantity

要在一場比賽中建立的此類型的隊伍數量。數量大於 1 的專案團隊會附加一個數字 (「Red_1」、「Red_2」等)。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

規則語句的集合，用於定義如何評估比賽的玩家。

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。規則名稱會在追蹤與規則相關活動的事件記錄檔和量度中參照。

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用途。它不會在配對過程中使用。

允許的值：字串

是否為必要？ 否

`type`

規則陳述式的類型。每個規則類型都有其他必須設定的屬性。如需每種規則類型結構和使用方式的詳細資訊，請參閱FlexMatch規則類型。

允許的值：

- 「AbsoluteSort」— 使用明確排序方法排序，該方法會根據指定的播放器屬性是否與批次中最舊的工單進行比較，以批次排序工單。
- 「收藏」— 評估集合中的值，例如收藏的玩家屬性，或多個玩家的一組值。
- 「比較」-比較兩個值。
- 「複合」— 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 人以下玩家的比賽。
- 「距離」— 測量數值之間的距離。
- 「批次距離」— 測量屬性值之間的差異，並使用它來分組比對請求。
- 「DispanceSort」— 使用明確的排序方法進行排序，該方法會根據具有數值的指定玩家屬性與批次中最舊票證的比較，在批次中排序票證。
- 「延遲」— 評估配對要求所報告的區域延遲資料。

是否為必要？ 是

`expansions`

在比賽無法完成時隨著時間的推移放寬比賽要求的規則。將資料片設定為逐步套用的一系列步驟，以便更容易找到相符項目。根據預設，FlexMatch會根據新增至比賽的最新票券的年齡計算等待時間。您可以使用 algorithm 屬性變更擴充等待時間的計算方式`expansionAgeSelection`。

擴充等待時間是絕對值，因此每個步驟的等待時間都應該比上一個步驟長。例如，若要排程逐步擴充系列，您可以使用 30 秒、40 秒和 50 秒的等待時間。等待時間不能超過匹配請求允許的最長時間，這是在配對配置中設置的。

是否為必要？ 否

target

要放鬆的規則集元素。您可以放鬆群組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，要更改團隊的最小規模：`teams[Red, Yellow].minPlayers`。若要變更名為「minSchle」的比較規則陳述式中的最低技能需求，請執行下列動作：`rules[minSkill].referenceValue`

是否為必要？ 是

steps

waitTimeSeconds

在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
```

批次距離規則可測量兩個屬性值之間的差異。您可以將批次距離規則類型用於大型和小型相符項。批次距離規則有兩種類型：

- 比較數字屬性值。例如，這種類型的批次距離規則可能要求一場比賽中的所有玩家都在彼此的兩個技能等級之內。對於此類型，請定義所有票證之間`batchAttribute`的最大距離。
- 比較字串屬性值。舉例來說，這種類型的批次距離規則可能會要求比賽中的所有玩家都要求相同的遊戲模式。針對此類型，定義FlexMatch用來形成批次的`batchAttribute`值。

批次距離規則性質

- **batchAttribute**— 用來形成批次的播放程式屬性值。

- **maxDistance**— 成功匹配的最大距離值。用於比較數值屬性。
- **partyAggregation**— 決定如何FlexMatch處理與多個玩家（派對）的彩票的價值。有效選項包括彩票玩家的最小值 (max)、最大值 (avg) 和平均值 ()。min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：

- 與參考值進行比較。例如，這種類型的比較規則可能要求匹配的玩家具有一定的技能等級或更高。對於此類型，請指定播放程式屬性、參照值和比較操作。
- 跨玩家進行比較。例如，這種類型的比較規則可能要求比賽中的所有玩家都使用不同的角色。針對此類型，請指定播放程式屬性，以及等於 (=) 或不等於 (!=) 比較運算。請勿指定參考值。

> **Note**
>
> 批次距離規則在比較玩家屬性時更有效率。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**— 將測量值與預期匹配進行比較。
- **operation**— 決定如何將量測與參照值進行比較的值。有效的作業包括：<<=、=、、！=、>、、>=。
- **partyAggregation**— 決定如何FlexMatch處理與多個玩家（派對）的彩票的價值。有效選項包括彩票玩家的最小值 (max)、最大值 (avg) 和平均值 ()。min預設值為 avg。

距離規則

```
distance
```

距離規則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，距離規則可能要求所有玩家玩遊戲至少 30 小時。

> **Note**
>
> 批次距離規則在比較玩家屬性時更有效率。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**— 測量距離的玩家屬性值。這必須是具有數值的屬性。
- **referenceValue**— 用於測量預期匹配距離的數值。
- **minDistance/maxDistance**— 成功比對的最小或最大距離值。
- **partyAggregation**— 決定如何FlexMatch處理與多個玩家（派對）的彩票的價值。有效選項包括彩票玩家的最小值 (max)、最大值 (avg) 和平均值 ()。min預設值為 avg。

收集規則

```
collection
```

收集規則會將一組玩家屬性值與批次中其他玩家的屬性值或參考值進行比較。一個集合可以包含多個玩家的屬性值，一個玩家屬性作為字符串列表，或兩者。例如，收集規則可能會查看團隊中玩家所選擇的角色。然後，該規則可能會要求團隊至少具有某個特定角色中的一個。

集合規則屬性

- **measurements**— 要比較的玩家屬性值的集合。屬性值必須是字串清單。
- **referenceValue**— 值 (或值的集合) 用來比較測量結果是否有潛在的比對。
- **operation**— 決定如何比較量測集合的值。有效的作業包括下列項目：
  - intersection— 此操作測量所有玩家收藏中相同的值的數量。如需使用交集作業的規則範例，請參閱範例 4：使用明確排序來尋找最符合的項目。
  - contains— 此作業會測量包含指定參照值的玩家屬性集合數量。如需使用「包含」作業的規則範例，請參閱範例 3：設定團隊層級需求和延遲限制。
  - reference_intersection_count— 此作業會測量玩家屬性集合中符合參照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播放器屬性。如需比較多個玩家屬性集合的規則範例，請參閱範例 5：尋找涵跨多個玩家屬性的交集。
- **minCount/maxCount**— 成功比對的最小或最大計數值。
- **partyAggregation**— 決定如何FlexMatch處理與多個玩家（派對）的彩票的價值。對於這個值，你可以用union來組合隊伍中所有玩家的玩家屬性。或者，您可以使intersection用該隊伍共同擁有的玩家屬性。預設值為 union。

複合規則

```
compound
```

複合規則使用邏輯語句來形成 40 個或更少玩家的匹配項。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。

您無法使用擴充規則來展開複合規則，但可以擴充基礎或支援規則。

複合規則屬性

- **statement**— 用來組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先前在規則集中定義。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— 如果提供的兩個引數為真，則表示式為 true。
  - or— 如果提供的兩個引數中的任何一個為 true，則表示式為 true。
  - not— 反轉運算式中引數的結果。
  - xor— 如果只有其中一個引數為真，則表示式為 true。

Example 範例

以下示例根據他們選擇的遊戲模式匹配不同技能水平的玩家。

```
{
    "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。延遲規則會忽略任何延遲高於最大值的位置。玩家在至少一個位置的延遲值必須低於最大值，延遲規則才能接受延遲規則。您可以透過指定maxLatency屬性，將此規則類型與大型相符項目一起使用。

延遲規則屬性

- **maxLatency**— 位置可接受的最大延遲值。如果票證的延遲位置低於最大值，則票證與延遲規則不符。
- **maxDistance**— 每張票的延遲與距離參考值之間的最大值。
- **distanceReference**— 與工單延遲進行比較的延遲值。在距離參考值的最大距離範圍內的門票將導致比賽成功。有效選項包括最小值 (min) 和平均 (avg) 播放程式延遲值。
- **partyAggregation**— 決定如何FlexMatch處理與多個玩家（派對）的彩票的價值。有效選項包括彩票玩家的最小值 (max)、最大值 (avg) 和平均值 ()。min預設值為 avg。

> **Note**
>
> 佇列可以將遊戲工作階段置於與延遲規則不相符的區域中。如需佇列延遲原則的詳細資訊，請參閱建立播放程式延遲原則。

絕對排序規則

```
absoluteSort
```

絕對排序規則會根據指定的玩家屬性，與新增至批次中的第一張票相比，對一批配對票券進行排序。

絕對排序規則屬性

- **sortDirection**— 對配對票券進行排序的順序。有效的選項包括ascending和descending。
- **sortAttribute**— 要排序票券的玩家屬性。
- **mapKey**— 排序玩家屬性的選項（如果它是地圖）。有效的選項包含：
  - minValue— 具有最低值的鍵是第一個。
  - maxValue— 具有最高值的鍵是第一個。
- **partyAggregation**— 決定如何FlexMatch處理與多個玩家（派對）的彩票的價值。有效選項包括最小（min）玩家屬性，最大（max）玩家屬性，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等級排序玩家，並平均派對的技能等級。

```
{
    "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中的第一張彩票的距離，對一批配對票券進行排序。

距離排序規則屬性

- **sortDirection**— 對配對票券進行排序的方向。有效的選項包括ascending和descending。
- **sortAttribute**— 要排序票券的玩家屬性。
- **mapKey**— 排序玩家屬性的選項 (如果是地圖)。有效的選項包含：

  - `minValue`— 針對新增至批次的第一份工單，尋找值最低的金鑰。
  - `maxValue`— 針對新增至批次的第一份工單，尋找具有最高值的金鑰。
- **`partyAggregation`**— 決定如何FlexMatch處理與多個玩家（派對）的彩票的價值。有效選項包括彩票玩家的最小值 (`max`)、最大值 (`avg`) 和平均值 ()。`min`預設值為 `avg`。

FlexMatch屬性表示式

性質表示式可用於定義某些相符項目相關的性質。它們允許您在定義屬性值時使用計算和邏輯。屬性運算式通常會產生以下兩種形式之一：

- 個人玩家數據。
- 個別玩家資料的計算收集。

常見的配對屬性表示式

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

目標	Input	意義	輸出
辨識配對中的特定團隊：	`teams[red]`	紅隊	團隊
若要識別比賽中的一組特定隊伍：	`teams[red,blue]`	红队和蓝队	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>
在一場比賽中識別一組特定隊伍中的球員：	`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>>

屬性表示式範例

下表說明基於先前範例建置的一些屬性運算式：

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]`	紅隊和藍隊中所有球員的「技能」屬性，按隊伍分組	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>>

屬性運算式彙總

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

聚合	Input	意義	輸出
`sum`	List<number>	取得列表中所有數字的總和。	數字
`count`	List<?>	取得列表中所有元素的數量。	數字
`stddev`	List<number>	取得列表中所有數字的標準差。	數字
`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	數字
平均 (球隊 [*]. 玩家. 屬性 [技能])	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	數字

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	數字
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	數字

FlexMatch配對活動

GameLiftFlexMatch亞馬遜在處理時為每個配對票發出事件。您可以將這些事件發佈到 Amazon SNS 主題，如中所述設定FlexMatch事件通知。這些事件也會以近乎即時的方式發出至 Amazon CloudWatch 活動，並以最大的努力為基礎。

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

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

詳細信息：類型，門票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

詳細資料：類型、工單、接受逾時、接受必要、、、MatchId 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。遺失資料表示該玩家AcceptMatch尚未呼叫該玩家。

資源：ConfigurationArn

詳細信息：類型，門票，匹配 ID，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

詳細信息：類型，門票，接受，匹配 ID，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

詳細信息：類型，門票，匹配 ID，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

詳細信息：類型，門票ruleEvaluationMetrics，消息，匹配 ID，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

已取消配對票證。

資源：ConfigurationArn

詳細信息：類型，門票ruleEvaluationMetrics，消息，匹配 ID，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

詳細信息：類型，門票ruleEvaluationMetrics，消息，匹配 ID，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"
  }
}
```

FlexMatch 的安全性

雲端安全是 AWS 最重視的一環。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。

安全是 AWS 與您共同肩負的責任。如需有關如何在使用時套用共同責任模型的資訊FlexMatch，請參閱 Amazon 中的安全性GameLift。

亞馬遜GameLiftFlexMatch版本資訊和 SDK 版本

Amazon GameLift 版本說明提供與服務相關的新FlexMatch功能、更新和修正的詳細資訊。此頁面還包括亞馬遜 GameLift SDK 版本歷史記錄。

亞馬遜GameLift開發資源

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

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

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。