亚马逊 GameLift

FlexMatch 开发人员指南

版本

亚马逊云科技

亚马逊 GameLift: FlexMatch 开发人员指南

Table of Contents

什么是 Amazon GameLift FlexMatch？

GameLift FlexMatch 是多人游戏的可自定义对战服务。借助 FlexMatch，您可以构建一组自定义规则，定义多人游戏比赛的外观，并确定如何评估和选择每场比赛的兼容玩家。您还可以自定义匹配过程的关键方面以适应您的游戏，包括微调匹配算法。

FlexMatch 既可以作为 GameLift 游戏托管解决方案（包括实时服务器）和独立匹配服务。你可以实现 FlexMatch 作为独立功能，其中包含使用的游戏 peer-to-peer 架构或在本地或其他云计算解决方案上托管游戏服务器（包括 GameLift FleetIQ)。本指南提供了有关如何针对上述任何情况构建配对系统的详细信息。

FlexMatch 让您可以根据游戏要求灵活地设置配对优先级。例如，可以：

- 在比赛速度和质量之间找到平衡。设置比赛规则以快速找到足够好的比赛，或者让玩家等待更长时间才能找到最佳的比赛以获得最佳玩家体验。
- 根据匹配良好的球员或匹配良好的球队进行比赛。创建一场比赛，其中所有玩家都有相似的特征，例如技能或经验。或者，在每个球队的组合特征相似的情况下，形式匹配，即使单个球员的特征更加多样化。
- 优先考虑玩家延迟如何影响到比赛中。为一场比赛中的所有玩家设置强制延迟限制，确保比赛中的每个人都经历类似的延迟，或者两者都有类似的延迟。

准备好开始使用 FlexMatch 了吗？

适用于 step-by-step 有关使用 FlexMatch 启动和运行您的游戏的指南，请参阅以下主题：

- FlexMatch 与 GameLift 托管集成 (p. 7)
- GameLift FlexMatch 集成独立配对 (p. 6)

密钥 FlexMatch 功能

所有功能都提供以下功能 FlexMatch 场景，无论您使用 FlexMatch 作为独立服务还是使用 GameLift 游戏托管。

- 可自定义的玩家匹配。设计和建造媒人，以适应你为玩家提供的所有游戏模式。构建一组自定义规则以评估关键玩家属性 (如技能级别或角色) 和地理延迟数据，为您的游戏构建最匹配的玩家。
- 基于延迟的匹配。提供玩家延迟数据并创建要求比赛中的玩家具有类似响应时间的比赛规则。当玩家匹配池跨越多个地理区域时，此功能非常有用。
- Support 最多 200 位玩家的对战规模。使用为你的游戏定制的比赛规则创建最多 40 名玩家的比赛。使用匹配过程创建最多 200 名玩家的比赛，该流程使用简化的自定义匹配流程来保持玩家的等待时间可管理。
- 玩家接受。要求玩家在完成比赛和开始游戏会话之前选择参加提议的比赛。使用此功能启动自定义验收工作流程并报告玩家回复 FlexMatch 在为对战组建新的游戏会话之前。如果不是所有玩家都接受比赛，那么建议的比赛将失败，接受的玩家自动返回配对池。

  玩家派对支持。为要组队一起玩游戏的组队生成对战游戏。使用 FlexMatch 以便根据需要寻找更多玩家填充对战。
- 可扩展的匹配规则。在经过一定时间之后，没有找到成功的对战，逐渐放松对战要求。规则扩展可让你决定在哪里和何时放宽初始比赛规则，以便玩家可以更快地进入可玩游戏。
- 匹配回填。使用匹配的新玩家填充现有游戏会话中的空位。自定义申请新玩家的时间和方式，并使用相同的自定义比赛规则来寻找其他玩家。

FlexMatch GameLift 托管

对于使用 GameLift 托管的游戏， FlexMatch 提供以下附加功能。使用时，这些都可用 GameLift 托管自定义游戏服务器，或者在使用实时服务器时。使用 Amazon Elastic Compute Cloud (Amazon EC2) 资源托管的游戏 GameLift FleetIQ 必须将 FlexMatch 作为独立功能实现。

- 游戏会话放置。当成功进行比赛时， FlexMatch 自动向 GameLift 请求新的游戏会话放置。配对过程中生成的数据，包括玩家 ID 和团队任务，都会提供给游戏服务器，以便它可以使用这些信息开始比赛的比赛会话。 FlexMatch 然后传回游戏会话连接信息，以便游戏客户端可以加入游戏。为了最大限度地减少玩家在比赛中经历的延迟，游戏会话放置 GameLift 也可以使用区域玩家延迟数据（如果提供）。
- 自动对战回填。启用此功能后， FlexMatch 当新游戏会话以未填充的玩家老虎机开始时，自动发送匹配回填请求。您的配对系统以最少数量的玩家开始游戏会话放置过程，然后快速填满剩余的插槽。你不能使用自动回填来替换退出匹配游戏会话的玩家。

定价 GameLift FlexMatch

GameLift 按使用持续时间对实例收取费用，以及按传输的数据量计算带宽的费用。如果你在上托管你的游戏 GameLift 服务器， FlexMatch GameLift 的费用中包含使用情况。如果你在另一个服务器解决方案上托管游戏， FlexMatch 使用量是单独收费的。有关 GameLift 的费用和价格的完整列表，请参阅亚马逊 GameLift 定价.

有关计算托管游戏或使用 GameLift 进行配对的成本的信息，请参阅生成 GameLift 定价估算，其中介绍了如何将Amazon Pricing Calculator.

Amazon GameLift FlexMatch 的工作原理

本主题概述了 GameLift FlexMatch 服务，包括 FlexMatch 系统的核心组件以及它们的交互方式。

您可以将 FlexMatch 用于使用 GameLift 托管托管的游戏或使用其他托管解决方案的游戏。在 GameLift 服务器（包括实时服务器）上托管的游戏使用集成的 GameLift 服务自动定位可用的游戏服务器并开始比赛的游戏会话。使用 FlexMatch 作为独立服务的游戏，包括 GameLift FleetIQ，必须与现有的托管系统协调，以分配主机资源并开始比赛的游戏会话。

有关为游戏设置 FlexMatch 的详细指导，请参阅开始使用 FlexMatch (p. 6).

对战组件

FlexMatch 对战系统包括以下部分或全部组件。

GameLift 组件

这些是 GameLift 资源，用于控制 FlexMatch 服务如何为您的游戏执行匹配。它们是使用 GameLift 工具创建和维护的，包括控制台和AmazonCLI 或者，或者以编程方式使用Amazon适用于 GameLift 的 SDK。

- FlexMatch 配置（也称为媒人）— 媒人是一组配置值，用于为你的游戏自定义匹配过程。游戏可以有多个对战构建器，每个对战构建器可根据需要为不同游戏模式或体验配置 当你的游戏向 FlexMatch 发送配对请求时，它会指定要使用哪个媒人。
- FlexMatch 对战规则集— 规则集包含评估玩家是否有潜在比赛以及批准或拒绝所需的所有信息。规则集定义了比赛的团队结构，声明用于评估的球员属性，并提供了描述可接受比赛标准的规则。规则可以应用于单个玩家、团队或整个对战。例如，一个规则可能需要对战中的每位玩家选择相同的游戏地图，或者它可能需要所有团队具有相似的玩家技能平均值。

- GameLift 游戏会话队列（仅适用于带 GameLift 托管托管的 FlexMatch）— 游戏会话队列可查找可用的托管资源，并为对战游戏启动新的游戏会话。该队列的配置决定 GameLift 在何处查找可用托管资源，以及如何为对战选择最佳可用主机。

自定义组件

以下组件包括完整的 FlexMatch 系统所需的功能，您必须根据游戏的体系结构实施这些功能。

- 用于匹配的玩家界面— 这个界面使玩家能够加入比赛。它至少通过客户匹配服务组件发起匹配请求，并根据匹配过程的需要提供玩家特定的数据，例如技能水平和延迟数据。

  Note

  作为最佳做法，与 FlexMatch 服务的通信应该由后端服务来完成，而不是来自游戏客户端。
- 客户对战服务— 此服务将从玩家界面输入玩家加入请求，生成匹配请求，然后将其发送到 FlexMatch 服务。对于处理中的请求，它监控配对活动，跟踪匹配状态，并根据需要采取行动。根据您管理游戏中的游戏会话托管方式，此服务可能会将游戏会话连接信息返回给玩家。此组件使用Amazon使用 GameLift API 的 SDK 与 FlexMatch 服务进行通信。
- 匹配放置服务（仅适用于作为独立服务的 FlexMatch）— 此组件与您现有的游戏托管系统配合使用，找到可用的托管资源并开始新的比赛会话以进行比赛。组件必须获得匹配结果并提取开始新游戏会话所需的信息，包括比赛中所有玩家的玩家 ID、属性和团队分配。

FlexMatch 对战过程

本主题介绍了基本的配对场景以及各种游戏组件与 FlexMatch 服务之间的交互。

请求玩家对战

使用游戏客户端的玩家点击 “加入游戏” 按钮。此操作会使您的客户配对服务向 FlexMatch 发送配对请求。该请求标识了在满足请求时要使用的 FlexMatch 媒人。该请求还包括自定义媒人所需的玩家信息，例如技能水平、播放偏好或地理延迟数据。你可以为一个玩家或多个玩家提出配对请求。

向配对池添加请求

当 FlexMatch 收到配对请求时，它会生成一张配对票并将其添加到媒人的门票池中。票证会留在池中，直到它获得匹配或达到最大时间限制。您的客户配对服务会定期收到关于配对活动的通知，包括门票状态的变化。

建立对战

您的 FlexMatch 媒人持续对其池中的所有票证运行以下流程：

1. 媒人按门票年龄对游泳池进行排序，然后从最古老的门票开始建立潜在的比赛。
2. 媒人将第二张门票添加到潜在的比赛中，并根据您的自定义匹配规则评估结果。如果潜在的比赛通过评估，那么门票的玩家将被分配给团队。
3. 媒人按顺序添加下一张票，然后重复评估过程。当所有玩家老虎机都填满后，比赛就准备好了。

大型比赛（41 至 200 名玩家）的配对使用上述过程的修改版本，以便它可以在合理的时间范围内建立比赛。媒人不是单独评估每张门票，而是将预先排序的门票池划分为潜在的比赛，然后根据您指定的玩家特征来平衡每场比赛。例如，媒人可能会根据类似的低延迟位置对门票进行预先排序，然后使用赛后平衡来确保球队与球员技能均匀匹配。

报告对战结果

当找到可接受的匹配时，所有匹配的门票都会更新，并为每个匹配的门票生成成功的匹配活动。

- FlexMatch 作为一个独立服务：你的游戏会在成功的匹配赛事中获得比赛结果。结果数据包括所有对战玩家及其团队任务的列表。如果你的比赛请求包含玩家延迟信息，结果还会建议比赛的最佳地理位置。

- FlexMatch 使用 GameLift 托管解决方案：比赛结果会自动传递到 GameLift 队列进行游戏会话放置。媒人决定了哪个队列用于游戏会话放置。

开始比赛的游戏会话

在建议的比赛成功组成后，新的游戏会话就开始了。在为比赛设置游戏会话时，您的游戏服务器必须能够使用匹配结果数据，包括玩家 ID 和团队分配。

- FlexMatch 作为一个独立服务：您的自定义比赛放置服务从成功的匹配赛事件中获取比赛结果数据，并连接到现有的游戏会话放置系统以找到比赛的可用主机资源。找到主机资源后，匹配放置服务将与您现有的托管系统协调，以启动新的游戏会话并获取连接信息。
- FlexMatch 使用 GameLift 托管解决方案：游戏会话队列可为对战找到最佳可用游戏服务器。根据队列的配置方式，它会尝试使用成本最低的资源放置游戏会话以及玩家将在哪里体验低延迟（如果提供了玩家延迟数据）。成功放置游戏会话后，GameLift 服务会提示游戏服务器启动新的游戏会话，传递配对结果和其他可选游戏数据。

将玩家 Connect 到比赛

游戏会话开始后，玩家连接到会话，领取他们的团队分配，然后开始游戏。

- FlexMatch 作为一个独立服务：你的游戏使用现有的游戏会话管理系统向玩家提供连接信息。
- FlexMatch 使用 GameLift 托管解决方案：在成功放置游戏会话时，FlexMatch 将使用游戏会话连接信息和玩家会话 ID 更新所有匹配的票证。

设置 FlexMatch

GameLift FlexMatch 是Amazon服务，你必须拥有Amazon账户才能使用此服务。创建 Amazon 账户是免费的。了解您使用可以完成的任务的更多信息Amazon账户，请参阅入门Amazon.

如果您将 FlexMatch 与其他 GameLift 解决方案配合使用，请参阅以下主题：

- 设置 GameLift 托管和实时服务器的访问权限
- 使用 GameLift FleetIQ 设置在 Amazon EC2 上托管的访问权限

为 GameLift 设置您的账户

1. 获取账户。打开Amazon Web Services然后选择登录控制台. 按照提示创建新账户或登录现有账户。
2. 设置管理用户组。打开Amazon Identity and Access Management(IAM) 服务控制台，然后按照步骤创建或更新用户或用户组。IAM 管理对您的访问Amazon服务和资源。必须向使用 GameLift 控制台或调用 GameLift API 访问 FlexMatch 资源的所有人提供明确访问权限。有关如何使用控制台的详细说明（或 AmazonCLI 或其他工具）来设置用户组，请参阅创建 IAM 用户.
3. 将权限策略附加到您的用户或用户组。针对 的访问权限Amazon服务和资源是通过附加IAM 策略转换到用户或用户组。权限策略指定一组Amazon用户必须有权访问的服务和操作。

   对于 GameLift，您必须创建自定义权限策略并将其附加到每个用户或用户组。策略是一个 JSON 文档。使用下面的示例创建策略。

以下示例说明了具有所有 GameLift 资源和操作的管理权限的内联权限策略。您可以选择通过仅指定 FlexMATCH 特定的项目来限制访问权限。

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

开始使用 FlexMatch

使用本节中的资源可以帮助您开始使用 FlexMatch 构建对战系统。

主题

GameLift FlexMatch 集成独立配对

本主题概述了实施的完整集成过程 FlexMatch 作为独立的对战服务。如果您的多人游戏是使用托管的，请使用此流程 peer-to-peer、自定义配置的本地硬件或其他云计算基元。此过程也可用于 GameLift FleetIQ，这是一款针对在 Amazon EC2 上托管的游戏的托管优化解决方案。如果你使用托管游戏 GameLift 托管主机（包括实时服务器），请参阅FlexMatch 与 GameLift 托管集成 (p. 7).

开始集成之前，您必须拥有一个Amazon账户并为其设置访问权限 GameLift 服务。有关详细信息，请参阅设置 FlexMatch (p. 5)。与创建和管理相关的所有基本任务 GameLift FlexMatch 媒人和规则集可以使用亚马逊完成 GameLift 控制台，但你可能也想要

1. 创建 FlexMatch 对战规则集。您的自定义规则集提供了有关如何构建匹配的完整说明。在其中，您可以定义每个团队的结构和规模。您还提供了一组比赛必须满足才能生效的要求，其中 FlexMatch 用于在比赛中包括或排除玩家。这些要求可能适用于个人玩家。您还可以自定义 FlexMatch 规则集中的算法，例如建立最多有 200 名玩家的大型比赛。请参阅这些主题：

   - 构建 FlexMatch 规则集 (p. 12)
   - FlexMatch 规则集示例 (p. 21)
2. 设置对战游戏通知。使用通知进行跟踪 FlexMatch 配对活动，包括待处理的比赛请求的状态。这是用于提供拟议比赛结果的机制。由于对战请求是异步的，您需要通过某种方式跟踪请求状态。使用通知是首选选项。请参阅这些主题：

   - 设置 FlexMatch 事件通知 (p. 35)
   - FlexMatch 对战事件 (p. 61)
3. 设置 FlexMatch 对战配置。也称为媒人，该组件接收配对请求并对其进行处理。您可以通过指定规则集、通知目标和最大等待时间来配置匹配器。您还可以启用可选功能。请参阅这些主题：

   - 设计一个 FlexMatch 媒人 (p. 8)
   - 创建对战配置 (p. 9)
4. 建立客户对战服务。创建或扩展游戏客户端服务，该服务具有构建和发送配对请求的功能 FlexMatch. 要建立配对请求，该组件必须具有获取配对规则集所需的玩家数据的机制，以及区域延迟信息（可选）。它还必须具有为每个请求创建和分配唯一票证 ID 的方法。你也可以选择建立玩家接受工作流程，要求玩家选择加入拟议的比赛。该服务还必须监控配对活动以获取比赛结果并为成功的比赛启动游戏会话放置。请参阅以下主题：

   - Add FlexMatch 到游戏客户端 (p. 38)
5. 建立比赛安置服务。创建一种可与您现有的游戏托管系统配合使用的机制，以找到可用的托管资源并启动新的游戏会话以成功进行比赛。此组件必须能够使用对战结果信息来获取可用的游戏服务器并为对战启动新的游戏会话。您可能还需要实现一个工作流程来提出比赛回填请求，该工作流程使用配对来填补已经在运行的匹配游戏会话中的空缺位置。

FlexMatch 与 GameLift 托管集成

FlexMatch 可用于适用于自定义游戏服务器和实时服务器的托管 GameLift 托管。要为您的游戏添加 FlexMatch 对战，请完成以下任务。

- 设置对战构建器。对战构建器会接收玩家的对战请求并进行处理。它根据一组定义的规则将玩家分组，并为每个成功的对战游戏创建新的游戏会话和玩家会话。请按照以下步骤设置对战构建器：
  - 创建规则集。规则集可以让对战构建器了解如何构建有效的对战游戏。它指定团队结构，并指定如何评估玩家是否参加对战游戏。请参阅这些主题：
    - 构建 FlexMatch 规则集 (p. 12)
    - FlexMatch 规则集示例 (p. 21)
  - 创建游戏会话队列。队列查找每个对战游戏的最佳区域，并在该区域中创建新的游戏会话。使用现有队列或者为对战创建一个新队列。请参阅以下主题：
    - 创建队列
  - 设置通知 (可选)。由于对战请求是异步的，您需要通过某种方式跟踪请求状态。通知是首选选项。请参阅以下主题：
    - 设置 FlexMatch 事件通知 (p. 35)
  - 配置一个对战构建器。一旦您拥有了规则集、队列和通知目标，即可为您的对战构建器创建配置。请参阅这些主题：
    - 设计一个 FlexMatch 媒人 (p. 8)
    - 创建对战配置 (p. 9)
- 将 FlexMatch 集成到您的游戏客户端服务中。向游戏客户端服务添加功能来启动包含对战的新游戏会话。对战请求指定要使用的对战构建器并为对战提供必要的玩家数据。请参阅以下主题：
  - Add FlexMatch 到游戏客户端 (p. 38)
- 将 FlexMatch 集成到您的游戏服务器。向游戏服务器添加功能来启动通过对战创建的游戏会话。此类游戏会话的请求包括对战特定信息，其中包括玩家和团队分配。在为对战构建游戏会话时，游戏服务器需要访问和使用这些信息。请参阅以下主题：
  - 将 FlexMatch 添加到 GameLift 托管的游戏服务器 (p. 42)
- 设置 FlexMatch 回填 (可选)。请求更多玩家匹配来填充现有游戏中空闲的玩家位置。您可以打开自动回填来让 GameLift 管理回填请求。或者，您可以通过向游戏客户端服务或游戏服务器添加功能以发起对战回填请求来手动管理回填。请参阅以下主题：
  - 使用 FlexMatch 回填现有游戏 (p. 43)

    Note

    回填当前对使用实时服务器的游戏不可用。

正在建一个 GameLift FlexMatch 媒人

一个 FlexMatch 对战构建器进程可用于构建对战游戏。它可以管理接收的对战请求池、组建参加对战游戏的团队、处理和选择玩家以找到最佳玩家组，以及启动为对战游戏放置和启动游戏会话的进程。本主题介绍对战构建器的主要方面以及如何为您的游戏配置一个自定义对战构建器。

有关 FlexMatch 对战构建器处理它收到的对战构建器处理它收到的对战请求FlexMatch 对战过程 (p. 3).

主题

设计一个 FlexMatch 媒人

本主题提供有关如何设计适合游戏的对战构建器的指导。

配置基本对战构建器

对战构建器至少需要具备以下元素：

- 规则集可确定对战团队的规模和范围并定义用于评估玩家是否参加对战的规则集。每个对战构建器均配置为使用一个规则集。请参阅 构建 FlexMatch 规则集 (p. 12) 和 FlexMatch 规则集示例 (p. 21)。
- 这些区域有：通知目标接收所有对战事件通知。您需要设置 Amazon Simple Notification Service (SNS) 主题，然后将主题 ID 添加到对战构建器中。有关设置通知的更多信息，请参阅设置 FlexMatch 事件通知 (p. 35)。
- 请求超时可确定对战请求留在请求池中以及被评估为潜在对战游戏的时长。一旦请求超时，则无法进行对战，并将从池中删除。
- 使用时 FlexMatch 和 GameLift 托管托管，游戏会话队列找到为对战游戏主持游戏会话的最佳可用资源，并启动新的游戏会话。每个队列都配置了Amazon用于确定可以放置游戏会话的位置的区域和资源类型（包括竞价型或按需实例）。有关队列的更多信息，请参阅。使用多区域队列.

选择一个Amazon对战构建器的区域

决定要在哪里进行配对活动，并在该区域创建你的媒人（配对配置和规则集）。对媒人的所有请求都被发送到那里的门票池，在那里对它们进行排序和评估，以确定可行的匹配。一旦进行比赛，玩家就可以被定向到受托管解决方案支持的任何位置的游戏会话。

选择时Amazon对于你的媒人来说，请考虑地点可能对他们的表现产生何种影响，以及如何为预期玩家优化比赛体验。我们建议使用以下最佳实践：

- 将媒人放在Amazon靠近玩家的区域以及发送的客户服务 FlexMatch 请求对战。这种方法减少了对配对请求工作流程的延迟影响，并提高了效率。
- 如果你的游戏覆盖全球受众，请考虑在多个地区创建媒人，并将比赛请求发送给离玩家最近的媒人。除了提高效率之外，这还会导致彩票池与地理位置相邻的玩家形成，从而提高了媒人根据延迟要求匹配玩家的能力。
- 使用时 FlexMatch 和 GameLift 托管托管，将您的对战构建器和使用的游戏会话队列放在同一个位置中。Amazon区域。这有助于最大程度地减少对战构建器和队列之间的通信延迟。

这些区域有： FlexMatch 资源MatchmakingConfiguration和MatchmakingRuleSet可以放在以下 GameLIFT 支持Amazon地区：美国东部 (弗吉尼亚北部)、美国西部 (俄勒冈)、中国 (北京和东京) 以及中国 (北京和东京) 以及中国 (北京和宁夏地区)。

添加可选元素

除了这些最低要求，您还可以为对战构建器配置以下附加选项。如果您使用的是 FlexMatch 用 GameLift 托管解决方案，内置了许多功能。如果您使用 FlexMatch 作为独立的配对服务，您可能需要在系统中构建这些功能。

玩家接受

您可以将对战构建器配置为要求选定参加对战的所有玩家都必须接受参与游戏。如果您的系统需要接受，所有玩家都必须选择接受或拒绝建议的对战游戏。对战游戏必须收到建议对战游戏的所有玩家的接受信息，才能完成。如果任何玩家拒绝或未接受对战游戏，建议的对战游戏将会丢弃建议的对战游戏，并按如下方式处理票证。对于所有玩家接受对战游戏的票证，将返回到对战池继续处理。如果至少有一个玩家拒绝或未能响应的对战游戏，将不再处理。玩家接受需要设置时间限制；所有玩家都必须在限制时间内接受建议的对战游戏才能继续对战。

回填模式

使用 FlexMatch 在游戏会话的整个生命周期内，让游戏会话始终有良好匹配的新玩家。在处理回填请求时，FlexMatch 使用与匹配原始玩家相同的对战构建器。您可以自定义回填门票与新比赛门票的优先顺序的方式，将回填票放到行的前端或末尾。这意味着，当新玩家进入对战池时，他们被放到现有游戏中的可能性要高于新组建的游戏。

无论你的游戏使用，都可以手动回填 FlexMatch 使用托管 GameLift 托管或其他托管解决方案。手动回填让您能够灵活地决定何时触发回填请求。例如，您可能希望仅在游戏的某些阶段添加新玩家。

自动回填仅适用于使用托管的游戏 GameLift 托管。启用此功能后，如果游戏会话以打开的玩家老虎机开始，GameLift 开始自动为它生成回填请求。此功能允许您设置配对，以便使用最少数量的玩家开始新游戏，然后在新玩家进入配对池时迅速填充。在游戏会话生命周期内，您可以随时关闭自动回填。

游戏属性

对于使用的游戏 FlexMatch 和 GameLift 托管托管，您可以在请求新的游戏会话时提供额外信息以传递给游戏服务器。这可能是一种有用的方法，可以通过游戏模式配置，这些配置是启动所创建的比赛类型所需的游戏模式配置。对战构建器创建的对战游戏会话将获得相同的游戏属性集。您可以通过创建不同的配对配置来更改游戏属性信息。

预留玩家位置

您可以指定为每个对战游戏预留的特定玩家位置，然后在日后占用这些位置。这可以通过配置对战配置的“额外玩家数量”属性完成。

自定义事件数据

使用此属性可在对战构建器的所有对战相关事件中包含一组自定义信息。此功能可用于跟踪您游戏独有的特定活动，包括跟踪对战构建器的性能。

创建对战配置

设置 GameLift FlexMatch 媒人来处理对接会要求, 创建一个对接会组态. 使用任一 GameLift 控制台或 Amazon Command Line Interface(Amazon CLI)。有关创建对战构建器的更多信息，请参阅设计一个 FlexMatch 媒人 (p. 8).

为创建对战构建器 GameLift 托管

在创建配对配置之前，创建规则集 (p. 20)和一个 GameLift 游戏会话队列与媒人一起使用。

Console

1. 打开 GameLift 控制台位于https://console.aws.amazon.com/gamelift/home.
2. 切换到Amazon要在其中创建对战构建的区域。有关支持的区域的列表， FlexMatch 配对配置，请参阅选择一个Amazon对战构建器的区域 (p. 8).
3. 在导航窗格中，选择FlexMatch、对战配置.
4. 在存储库的对战配置页面上，选择创建配置.
5. 在存储库的定义配置详细信息。页面，位于对战配置详细信息。中，执行以下操作：

   a. 适用于名称，请输入一个对战构建器名称，以帮助您在列表和指标中识别对战构建器。对战构建器名称在区域中必须唯一。对战请求会使用其名称和区域标识要使用的对战构建器。

   b. （可选）对于说明，添加描述以帮助标识对战构建器。

   c. 适用于规则集中，从列表中选择要与媒人一起使用的规则集。该列表包含您在当前区域中已创建的所有规则集。

   d. 适用于FlexMatch 模式，选择托管为了 GameLift 管理的托管。此模式提示 FlexMatch 将成功的匹配传递到指定的游戏会话队列。

   e. 适用于Amazon区域，请选择待对战构建器使用的游戏会话队列配置的区域。

   f. 适用于队列，请选择要与对战构建器一起使用的游戏会话队列。
6. 选择 Next（下一步）。
7. 在存储库的配置设置页面，位于对战设置中，执行以下操作：

   a. 适用于请求超时中，设置对战构建器针对每个请求完成对战游戏的最长时间（以秒为单位）。超过该时间的对战请求都将拒绝。

   b. 适用于回填模式中，选择处理对战回填的模式。打开自动回填功能，选择自动. 或者，如果您在游戏服务器或游戏客户端管理回填请求，或者您选择不回填游戏，请选择手动.

   c. （可选）对于其他玩家数，设置在匹配中保持打开的玩家位置数。 FlexMatch 将future 玩家可以占用这些位置。

   d. （可选）在对战接受选项，对于需要接受，如果您希望建议对战游戏中的每个玩家主动接受参与对战游戏，请选择必需. 如果选择此选项，则对于接受超时，设置您希望对战构建器在等待玩家接受多长时间（以秒为单位），表示您希望对战构建器在等待玩家接受多长时间（以秒为单位）。
8. （可选）在事件通知设置中，执行以下操作：

   a. （可选）对于SNS 主题, 选择Amazon Simple Notification Service (Amazon SNS) 接收主题对战事件通知。如果您尚未设置 SNS 主题，可以在以后通过编辑对战配置来选择此主题。有关更多信息，请参阅 设置 FlexMatch 事件通知 (p. 35)。

   b. （可选）对于自定义事件数据中，输入要与该对战构建器关联的、事件消息中的任何自定义数据。 FlexMatch 在与对战构建器关联的每个事件中包含此数据。
9. (可选)其他游戏数据中，然后执行以下操作：

   a. （可选）对于游戏会话数据中，输入您想要的任何其他游戏相关信息 FlexMatch 要交付给新的游戏会话，首先是使用此配对组成的比赛.

   b. （可选）对于游戏属性中，添加包含有关对战构建器会话信息的键值对战属性。
10. （可选）在标签，添加标签以帮助你管理和跟踪你的Amazon资源的费用。
11. 选择 Next（下一步）。
12. 在存储库的审核和创建页面上，查看您的选择，然后选择Create. 成功创建后，对战构建器会准备好接受对战请求。

Amazon CLI

要使用 Amazon CLI 创建对战配置，请打开命令行窗口，然后使用 create-matchmaking-configuration 命令定义一个新对战构建器。

该示例命令会创建一个新的对战配置，要求玩家接受和启用自动回填。此外，它还预留两个玩家位置 FlexMatch 以便稍后添加玩家，它提供了一些游戏会话数据。

```
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"
```

如果对战配置创建请求成功， GameLift返回MatchmakingConfiguration对象，其中包含为对战构建器请求的设置。对战构建器会准备好接受对战请求。

为独立创建媒人 FlexMatch

在创建配对配置之前，创建规则集 (p. 20)与媒人一起使用。

Console

1. 打开 GameLift 控制台位于https://console.aws.amazon.com/gamelift/home.
2. 切换到Amazon要在其中创建对战构建的区域。有关支持的区域的列表， FlexMatch 配对配置，请参阅选择一个Amazon对战构建器的区域 (p. 8).
3. 在导航窗格中，选择FlexMatch、对战配置.
4. 在存储库的对战配置页面上，选择创建配置.
5. 在存储库的定义配置详细信息。页面，位于对战配置详细信息。中，执行以下操作：
   a. 适用于名称，请输入一个对战构建器名称，以帮助您在列表和指标中识别对战构建器。对战构建器名称在区域中必须唯一。对战请求会使用其名称和区域标识要使用的对战构建器。
   b. （可选）对于说明，添加描述以帮助标识对战构建器。
   c. 适用于规则集中，从列表中选择要与媒人一起使用的规则集。该列表包含您在当前区域中已创建的所有规则集。
   d. 适用于FlexMatch 模式，选择独立的. 这表示你有一个自定义机制，可以在托管解决方案之外启动新的游戏会话 GameLift.
6. 选择 Next（下一步）。
7. 在存储库的配置设置页面，位于对战设置中，执行以下操作：
   a. 适用于请求超时中，设置对战构建器针对每个请求完成对战游戏的最长时间（以秒为单位）。超过该时间的对战请求都将拒绝。
   b. （可选）在对战接受选项，对于需要接受，如果您希望建议对战游戏中的每个玩家主动接受参与对战游戏，请选择必需. 如果选择此选项，则对于接受超时，设置您希望对战构建器在等待玩

家接受多长时间（以秒为单位），表示您希望对战构建器在等待玩家接受多长时间（以秒为单位）。

8. （可选）在事件通知设置中，执行以下操作：

   a. （可选）对于SNS 主题，请选择一个 Amazon SNS 主题以接收对战事件通知。如果您尚未设置 SNS 主题，可以在以后通过编辑对战配置来选择此主题。有关更多信息，请参阅 设置 FlexMatch 事件通知 (p. 35)。

   b. （可选）对于自定义事件数据中，输入要与该对战构建器关联的、事件消息中的任何自定义数据。 FlexMatch 在与对战构建器关联的每个事件中包含此数据。

9. （可选）在标签，添加标签以帮助你管理和跟踪你的Amazon资源的费用。

10. 选择 Next（下一步）。

11. 在存储库的审核和创建页面上，查看您的选择，然后选择Create. 成功创建后，对战构建器会准备好接受对战请求。

Amazon CLI

要使用 Amazon CLI 创建对战配置，请打开命令行窗口，然后使用 create-matchmaking-configuration 命令定义一个新对战构建器。

此示例命令为需要玩家接受的独立媒人创建一个新的配对配置.

```
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"
```

如果对战配置创建请求成功， GameLift返回MatchmakingConfiguration对象，其中包含为对战构建器请求的设置。对战构建器会准备好接受对战请求。

构建 FlexMatch 规则集

每个 FlexMatch 对战构建器必须拥有一个规则集。规则集可确定对战游戏的两个关键元素：游戏团队的结构和规模，以及如何为玩家分组才能呈现可能的最佳对战游戏。

例如，规则集可能会描述这样的匹配项：创建包含两个团队的对战游戏，每个团队有 5 个玩家，一个团队是防守者，另一个团队是进攻者。一个团队可以有新手玩家和经验丰富的玩家，但两个团队的平均技能必须在 10 点内。如果在 30 秒后未能进行匹配，则逐渐放宽技能要求。

本部分的主题介绍如何设计和构建对战规则集。在创建规则集时，您可以使用 Amazon GameLift 控制台或 AmazonCLI。

主题

- FlexMatch 规则语言 (p. 49)

设计 FlexMatch 规则集

在最基本的层面上，对战规则集必须有两个作用：第一，安排对战游戏的团队结构和规模；第二，告知对战构建器如何选择玩家以构建可能的最佳匹配。

但是，您的对战规则集可以做的更多。例如，您可以：

- 自定义匹配算法以针对游戏进行优化。
- 设置最低玩家延迟要求来保护游戏质量。
- 随着时间的推移，逐渐放松球队要求和/或比赛规则，以帮助确保所有活跃玩家在想要的时候找到可接受的比赛。
- 为包含多个玩家（各方聚合）的对战请求定义特殊处理。
- 为大型对战（> 40 位玩家）触发特殊的匹配处理。在 设计 FlexMatch 大型匹配规则集 (p. 17) 中了解有关构建大型对战游戏的更多信息。

本主题介绍规则集的基本结构以及如何为小型对战构建规则集（最多 40 位玩家）。有关其他帮助，请参阅以下 FlexMatch 规则参考主题：

- 创建对战规则集 (p. 20)
- FlexMatch 规则集示例 (p. 21)
- FlexMatch 规则语言 (p. 49)
- FlexMatch 规则集架构 (p. 49)

在构建对战规则集时，您可以执行以下部分或全部任务：

- 描述规则集 (p. 13)(必需)
- 自定义匹配算法 (p. 13)（可选）
- 声明玩家属性 (p. 16)
- 定义对战团队 (p. 16)
- 设置玩家匹配规则 (p. 17)
- 随着时间的推移允许要求放松 (p. 17)

需要设置规则集来构建对战有 40 位以上玩家？ 了解如何设计 FlexMatch 大型匹配规则集 (p. 17)。

描述规则集

提供规则集的详细信息。

- 名称（可选）— 这是供您自己使用的描述性标签，不能供 Amazon GameLift 以任何有意义的方式使用。此值与您在使用 GameLift 创建规则集时指定的规则集名称没有关联。
- 规则语言版本（必需）— 这是用于创建 FlexMatch 规则的属性表达式语言的版本。此值必须等于“1.0”。

自定义匹配算法

您可以选择修改默认匹配算法中的某些元素。根据设计，默认算法针对大多数游戏进行了优化，以便快速高效地处理比赛请求，让玩家以最短的等待时间进入可接受的比赛 您可以自定义算法并调整对战优先级，以更好地满足您的游戏的需求。

默认的 FlexMatch 配对流程如下：

1. 所有开放的配对门票和回填门票都放在门票池中。
2. 池中的票证随机分组为一个或多个批次进行匹配。随着票证池的增大，会形成更多批次，以确保批次保持最佳大小以进行匹配。
3. 在每批次中，门票按年龄排序。
4. 匹配是围绕每批中最早的门票构建的，评估批次中的所有其他门票以找到可接受的匹配。

要自定义匹配算法，请添加`algorithm`组件到规则集模式中。请参阅FlexMatch 规则集架构 (p. 49)了解完整的参考信息。

使用以下可选自定义项来影响配对过程的不同阶段。

- 添加预批量分类 (p. 14)
- 优先顺序填充票证 (p. 15)
- 通过扩展支持较旧的机票 (p. 15)

添加预批量分类

您可以配置 FlexMatch 以在形成批处理之前对票证池进行排序。这种类型的自定义对于往往拥有非常大的门票池的游戏来说最有效。预批量分拣可以帮助加快配对过程。它还倾向于在某些特征上形成具有更大的球员均匀性的匹配。

在算法属性中定义了预批量排序方法`batchingPreference`. 默认设置为 “随机”，表示不会进行预先排序。

自定义批量预排序的选项包括以下内容：

- 按玩家属性排序。提供玩家属性列表以对票池进行预先排序。这种自定义使 FlexMatch 能够在排序的属性中创建具有更大一致性的批次。例如，如果你按玩家技能对票池进行预先排序，那么具有相似技能水平的门票往往会被批处理在一起。如果你的规则集还包含基于玩家技能的比赛规则，那么预批量排序可以显著提高配对效率。

  要启用此自定义功能，请设置算法属性`batchingPreference`为 “已排序”，然后设置属性`sortByAttributes`到玩家属性列表中。列表中的每个属性都要在`playerAttributes`设置规则集的组成部分。

  在以下示例中，FlexMatch 首先根据玩家的首选游戏地图，然后按玩家技能对票池进行排序。生成的批次更有可能包含想要使用同一张地图的类似熟练的玩家。

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

- 按延迟排序。选择以下选项之一的优先级：(1) 让玩家参加可用延迟最低的比赛，或 (2) 以可接受的延迟快速让玩家进入比赛。这种自定义适用于组建大型比赛的规则集（超过 40 名玩家）；算法属性`strategy`必须设置为 “平衡”，这大大限制了可用的规则声明类型。请参阅 设计 FlexMatch 大型匹配规则集 (p. 17)。

  此自定义使 FlexMatch 可以通过以下方式之一根据报告的延迟数据对票证进行预排序：

  - 让玩家进入延迟最低的区域。门票池按玩家报告其最低延迟值的地区进行预先排序。这导致 FlexMatch 批量报告相同地区低延迟的门票，这往往会将玩家匹配到他们最快的区域和整体上更好的游戏体验。它还减少了每批票的数量，因此完成比赛可能需要更长的时间。要启用此自定义功能，请设置算法属性`batchingPreference`将值转为 “FastRegion”，如以下示例所示。

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

- 快速让玩家进入可接受的延迟比赛。票务调查按玩家报告任何可接受的延迟值的地区进行预先排序。这种方法往往形成较少的批次，每个批次都包含大量在同一地区具有可接受延迟的票证。随着每批门票的更多，找到足够可接受的匹配往往会更容易、更快捷。要启用此自定义功能，请设置属性`batchingPreference`将值转为 “LargestPules”，如以下示例所示。（此方法是使用平衡策略的规则集的默认行为。）

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

优先顺序填充票证

如果您的游戏实现了自动回填或手动回填，则可以根据请求类型（新匹配或回填请求）自定义 FlexMatch 处理配对票的方式。默认情况下，两种类型的请求都被平等对待。这种自定义会导致 FlexMatch 要么首先尝试填写回填票，要么仅在无法进行新匹配的情况下填写票证。它还有可能影响托管资源的使用 —— 要么将玩家打包到更少的游戏会话中，或者通过创建更部分填充的游戏会话。

回填优先级排序会影响票证在批处理票证后的处理方式；它不会影响 TT 批处理过程。您可以根据需要实施批量预分拣和回填优先级。仅对使用详尽搜索策略的规则集使用回填优先级。

要更改回填票的优先级，请设置属性`backfillPriority`如下所示：

- 首先匹配回填票。此选项提示 FlexMatch 在创建新匹配之前评估并尝试完成回填票证。这意味着即将进入的玩家有更高的机会被插入现有游戏。设置属性`backfillPriority`变为 “高”。

  如果你的游戏使用自动回填，请启用此自定义作为最佳实践。自动回填最常用于游戏会话时间较短且玩家周转率高的游戏中。自动回填有助于这些游戏快速形成最低可行的比赛并开始比赛，即使 FlexMatch 搜索更多玩家以填补空缺的老虎机。首先尝试填写回填单有助于优化这种方法。

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

- 最后匹配回填票。此选项会提示 FlexMatch 忽略回填票证，直到评估了所有其他票证。这意味着，只有在无法匹配到新游戏的时候，才会将其回填到现有游戏中。设置属性`backfillPriority`变为 “低”。

  当你想使用回填作为让玩家进入游戏的最后机会选项时，例如当玩家太少无法组建新比赛时，此选项非常有用。取消回填票的优先级不应与自动回填一起使用。

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

通过扩展支持较旧的机票

自定义 FlexMatch 如何应用扩展规则，在比赛难以完成时放宽匹配标准。当部分完成的比赛中的门票达到一定年龄时，GameLift 应用扩展规则。票证的创建时间戳决定何时应用规则；默认情况下，FlexMatch 会跟踪最近匹配票证的时间戳。

要更改应用扩展规则的时间，请设置属性`expansionAgeSelection`如下所示：

- 根据最新票证扩展。此选项根据添加到潜在比赛中的最新票证来应用扩展规则。每次匹配新票时，时钟都会重置。有了这个选项，应用扩张更加困难；结果的比赛质量往往更高，但是玩家的等待时间可能会更长，并且比赛请求可能会在完成之前超时。设置属性`expansionAgeSelection`改为 “最新”（这是默认值）。
- 根据最旧的票证进行扩展。此选项应用基于潜在比赛中最早的门票的扩展规则，该门票通常是（但并非总是）添加到比赛中的第一张门票。有了这个选项，扩张通常会更快地应用，这可以缩短最早匹配球员的等待时间，但也降低了所有玩家的比赛质量。设置属性`expansionAgeSelection`到 “最古老的”。

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

声明玩家属性

在本节中，列出将包含在配对请求中的单个玩家属性。您可能需要在规则集中声明玩家属性有两个原因：

- 当规则集包含依赖玩家属性的任何规则时。必须声明匹配规则中使用的所有属性。
- 当你想通过匹配请求将玩家属性传递给游戏会话时，即使该属性未在匹配过程中使用也是如此。例如，您可能希望在每个玩家连接之前将玩家角色选择传递给游戏会话。

在声明玩家属性时，请包含以下信息：

- 名称此值在规则集中必须唯一。
- 类型(必需)-属性值的数据类型。有效数据类型为数字、字符串或字符串映射。
- 默认（可选）— 如果对战请求未提供属性值，则输入要使用的默认值。如果未声明默认值，请求确实包含值，则无法满足该请求。

定义对战团队

描述对战游戏的团队的结构和规模。每个对战游戏必须至少有一个团队，您可以根据需要定义任意数量的团队。您的团队的玩家数量可以相同，也可以不同。例如，您可以定义有一个玩家的怪物团队和有 10 个玩家的猎人团队。

FlexMatch 将匹配请求处理为小型对战或大型对战，具体取决于规则集如何定义团队规模。最多 40 位玩家的潜在对战游戏属于小型对战，而超过 40 位玩家的对战游戏则属于大型对战。要确定规则集的潜在对战游戏规模，请为在规则集中定义的所有团队添加 maxPlayer 设置。

- 名称(必需)-为每个团队分配一个唯一名称。此名称在规则和扩展中使用，并在用于游戏会话的对战数据中引用。
- MaxPlayers(必需)-指定可以分配到团队的玩家的最大数量。
- minPlayers（必需）— 指定在对战成功前必须分配给团队的最少玩家数量。
- 数量（可选）— 如果您希望 FlexMatch 根据此定义创建多个团队，请指定数量。当 FlexMatch 创建对战游戏时，会为这些团队提供指定名称并附加一个编号。例如，“Red-Team_1”、“Red-Team_2”、“Red-Team_3”等。

FlexMatch 始终会尝试将团队填充到最大玩家规模，但在最小玩家规模允许的情况下，会创建团队，玩家数量较少。如果您希望对战游戏的所有团队都具有相同规模，可以创建相应的规则。有关“EqualTeamSizes”规则的示例，请参阅 FlexMatch 规则集示例 (p. 21)主题。

设置玩家匹配规则

创建一组定义如何评估玩家在对战游戏中的接受情况的规则语句。规则可以设置应用于单个玩家、团队或整个对战的要求。当 GameLift 处理对战请求时，它将从可用玩家池中最早的玩家开始，围绕该玩家构建对战。有关创建 FlexMatch 规则的详细帮助，请参阅FlexMatch 规则类型 (p. 55).

- 名称（必需）— 这是用于在规则集中唯一标识规则的有意义的名称。规则名称也可在跟踪与此规则相关的活动的事件日志和指标中引用。
- 描述(可选)-使用此元素可附加自由格式文本描述。此信息不可供对战构建器使用。
- 类型类型元素标识处理规则时要使用的操作。每个规则类型都需要一组额外属性。例如，若干规则类型需要一个引用值来衡量玩家的属性。有关有效的规则类型和属性的列表，请参阅 FlexMatch 规则语言 (p. 49)。
- 规则类型属性（可能是必需的）— 根据定义的规则类型，您可能需要设置某些规则属性。例如，对于距离和比较规则，必须指定要衡量的玩家属性。请在中了解有关属性以及如何使用 FlexMatch 属性表达式语言的更多FlexMatch 规则语言 (p. 49).

随着时间的推移允许要求放松

当对战无法完成时，扩展允许您随着时间的推移放宽匹配条件。此功能可确保在不可能有完美匹配时，进行“现有最佳”匹配。您可以使用扩展来放宽玩家技能要求，提高可接受的玩家延迟级别，或降低团队所需的最少玩家数量。通过放宽规则，您将逐渐扩大可接受的对战游戏的玩家池。

当未完成比赛中最新票的时间与扩展等待时间相匹配时间时，将触发扩展。当新票添加到比赛中时，扩展等待时间时钟可能会重置。您可以在`algorithm`设置规则集的部分。请记住，扩张步骤等待时间是绝对的，它们彼此不会复杂。因此，如果你有两个步骤，一个设置为 15，一个设置为 30，第二步将在第一步后 15 秒触发。

以下是一个扩张的例子，它逐渐提高了比赛所需的最低技能水平。规则集使用名为 “SkillDelta” 的距离规则语句来要求对战游戏的所有玩家在 5 个技能级别以内。如果 15 秒钟内没有进行新的比赛，则这种扩展允许技能水平差异 10，然后十秒后允许差异 20 秒。

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

在启用了自动回填的对战构建器时，不要太快地放宽玩家计数要求。新游戏会话需要几秒钟时间启动并开始自动回填。如果您的扩展步骤具有非常短的等待时间，在新启动的游戏会话发送回填请求前，FlexMatch 可能会创建很多部分填充的对战。更好的方法是仅在您的游戏开始自动回填后触发扩展。这有助于 FlexMatch 更快、更高效地为游戏（新游戏或现有游戏）获取玩家。扩展时间取决于团队构成，因此请进行一些测试来找到最适合您的游戏的扩展策略。

设计 FlexMatch 大型匹配规则集

如果您的规则集创建允许 41 到 200 位玩家的对战游戏，则需要对规则集配置作出一些调整。这些调整优化了比赛算法，以便它可以建立可行的大型比赛，同时还可以缩短玩家的等待时间。因此，大型匹配规则集用针对常见匹配优先级进行了优化的标准解决方案取代耗时的自定义规则。

以下是确定是否需要针对大型比赛优化规则集的方法：

1. 对于规则集中定义的每个团队，获取maxPlayer、
2. 将所有加起来maxPlayer值。如果总计超过 40，您将有大型对战规则集。

要针对大型比赛优化规则集，请按如下所述进行调整。请参阅中的大型匹配规则集架构大型匹配的规则集架构 (p. 51)和规则集示例示例 7：创建大型比赛 (p. 30).

为大型比赛自定义匹配算法

将算法组件添加到规则集（如果该组件不存在）。设置以下属性。

- `strategy`（必填）— 设置`strategy`财产改为 “平衡”。此设置触发 FlexMatch 进行额外的赛后检查，以根据指定的球员属性找到最佳球队余额，该属性在`balancedAttribute`财产。平衡策略取代了对自定义规则的需求，以建立均匀匹配的团队。
- `balancedAttribute`在对战中平衡团队时确定要使用的玩家属性。此属性必须具有数值数据类型（双精度或整数）。例如，如果你选择平衡球员技能，FlexMatch 会尝试分配球员，以便所有球队拥有尽可能均匀匹配的总技能水平。平衡属性必须在规则集的玩家属性中声明。
- `batchingPreference`（可选）— 选择你想强调多少重点来为玩家打造尽可能的最低延迟比赛。此设置影响匹配票在建立匹配之前的排序方式。选项包括：
  - 最大群体。FlexMatch 允许使用池中至少在一个共同区域具有可接受延迟值的所有票证。因此，潜在的门票池往往很大，这使得更快地填补比赛变得更容易。玩家可能会被放置在游戏中的延迟时间可以接受但并非总是最佳的。如果`batchingPreference`属性未设置，这是默认行为：`strategy`设置为 “平衡”。
  - 最快区域。FlexMatch 根据报告最低延迟值的位置对池中的所有票证进行预排序。因此，对战往会与报告在相同区域具有较低延迟的玩家组建。同时，每场比赛的潜在门票池较小，这可能会增加填补比赛所需的时间。而且，由于延迟更高，对战游戏的玩家在平衡属性方面可能变化更大。

以下示例将匹配算法配置为行为如下：(1) 对门票池进行预先排序，以便按具有可接受延迟值的区域对门票进行分组；(2) 形成批次排序的门票进行匹配；(3) 使用批次门票创建比赛并平衡球队以平衡玩家的平均技能。

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

声明玩家属性

请确保您声明在规则集算法中用作平衡属性的玩家属性。匹配请求中应包含每位玩家的此属性。您可以为玩家属性提供默认值，但是在提供玩家特定的值时，属性平衡效果最佳。

定义团队

定义团队规模和结构的过程与小型对战相同，但 FlexMatch 填充团队的方式不同。这将影响在只有部分填充时对战可能呈现的外观。您可能想要调整响应中的最小团队规模。

FlexMatch 在将玩家分配到团队时使用以下规则。首先：查找尚未达到其最低玩家要求的团队。其次：在这些团队中，查找具有最多空闲位置的团队。

对于定义多个同等规模团队的对战，玩家将按顺序添加到每个团队，直到填满。因此，对战游戏的团队始终拥有接近等数量的玩家，即使对战未填满。目前，还没有方法在大型对战中强制定义规模相同的团队。对于规模不对称的团队，过程稍微复杂一些。在这种情况下，玩家最开始会被分配给空闲位置最多的最大团队。当空闲位置的数量在所有团队之间更均匀地分配，玩家被划入更小的团队。

例如，假设您有一个包含三个团队的规则集。红色团队和蓝色团队均设置为`maxPlayers=10`,`minPlayers=5`. 绿色团队将设置为`maxPlayers=3``minPlayers=2`. 以下是填充顺序：

1. 没有团队已达到minPlayers. 红色团队和蓝色团队有 10 个空闲位置，绿色团队有 3 个。前 10 个玩家被分配（每个团队 5 个）到红色团队和蓝色团队。两个团队现在均已达到minPlayers.
2. 绿色团队尚未达到minPlayers. 接下来 2 个玩家被分配到绿色团队。绿色团队现在已经到达minPlayers.
3. 与所有团队一起minPlayers现在，根据空闲位置的数量分配额外玩家。红色和蓝队各有 5 个空缺槽，而绿队有 1 个。接下来的 8 个玩家将被分配（每个团队 4 个）到红色团队和蓝色团队。所有球队现在都有 1 个开放时隙。
4. 剩余的 3 个玩家位置将不按特定顺序分配（每个团队 1 个）到团队。

为大型比赛设置规则

大型比赛的配对主要依赖于平衡策略和延迟批处理优化。大多数自定义规则不可用。但是，您可以合并以下类型的规则：

- 对玩家延迟设置硬性限制的规则。使用latency具有属性的规则类型maxLatency. 请参阅延迟规则 (p. 57)引用。下面是一个将最大玩家延迟设置为 200 毫秒的示例：

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

- 规则根据指定玩家属性中的亲密度对玩家进行批处理。这不同于将平衡属性定义为大型比赛算法的一部分，后者侧重于构建均匀匹配的团队。该规则根据指定属性值（例如初学者或专家技能）的相似性对配门票进行分组，这往往会导致与指定属性紧密对齐的玩家进行比赛。使用batchDistance规则类型，标识基于数字的属性，并指定允许的最宽范围。请参阅Batch 规则规则 (p. 55)引用。以下是一个示例，要求比赛的球员在彼此的一个技能水平之内：

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

放松大型比赛要求

在小型对战中，您可以使用扩展来在可能没有有效匹配时随着时间推移放宽匹配要求。在大型对战中，您可以选择放宽延迟规则或团队玩家计数。

如果您在大型对战中使用自动匹配回填，请不要太快地放宽团队玩家计数。FlexMatch 仅在游戏会话启动后开始生成回填请求，在对战创建后的前几秒钟可能不会开始。在此期间，FlexMatch 可以创建多个部分填充的新游戏会话，尤其是在玩家计数规则降低时。因此，您最后将有多于所需的更多游戏会话，而且玩家将在这些会话之间过于稀疏地分布。最佳做法是给予玩家计数扩展的第一个步骤更长的等待时间，长到足够让游戏会话启动。由于为大型对战的回填请求提供了更高的优先级，传入玩家将在新游戏启动前被放入现有游戏。您可能需要进行实验来找到最适合您的游戏的等待时间。

下面是一个在更长的初始等待时间内逐渐降低黄色团队的玩家计数的示例。请记住，规则集扩展中的等待时间是绝对的，不是复合的。因此，第一次扩展在第五秒时进行，第二次扩展在五秒以后，即第十秒时进行。

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

```
            "value": 5
        }]
    }]
```

创建对战规则集

在创建您的对战规则集之前 GameLift FlexMatch 媒人，我们建议您查看规则集语法 (p. 49). 使用创建规则集后 GameLift 控制台或Amazon Command Line Interface(Amazon CLI)，无法更改此设置。

请注意，有一个服务配额以获取您可以在AmazonRegion，因此最好将不使用的规则集删除。

相关主题

- 设计 FlexMatch 规则集 (p. 13)
- FlexMatch 规则集示例 (p. 21)
- FlexMatch 规则语言 (p. 49)

Console

创建规则集

1. 打开 GameLift 控制台位于https://console.aws.amazon.com/gamelift/.
2. 切换到Amazon要在其中创建规则集的区域。在使用该规则集的对战配置所在的区域中定义该规则集。
3. 在导航窗格中，选择FlexMatch、对战规则集.
4. 在存储库的对战规则集页面中，选择创建规则集.
5. 在存储库的创建对战规则集页面中，执行以下操作：
   a. UNDER规则集设置，对于名称中，输入一个可用于在列表或事件和量度表中标识该名称的唯一描述性名称。
   b. 适用于规则集中，以 JSON 格式输入您的规则集。有关设计规则集的信息，请参阅设计 FlexMatch 规则集 (p. 13). 您还可以使用以下示例规则集FlexMatch 规则集示例 (p. 21).
   c. 选择验证以验证您的规则集语法正确。规则集在创建之后无法编辑，因此先验证这些规则集是一种好做法。
   d. （可选）在标签，添加标签以帮助你管理和跟踪你的Amazon资源的费用。
6. 选择 Create（创建）。如果创建成功，您可以将该规则集与对战构建器结合使用。

Amazon CLI

创建规则集

打开命令行窗口，然后使用命令create-matchmaking-rule-set.

此示例命令创建设置单个团队的简单对战规则集。请确保在同一个规则集中Amazon区域作为使用它的配对配置.

```
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}]}'
```

如果创建请求成功， GameLift 返回MatchmakingRuleSet对象，其中包含您指定的设置。媒人现在可以使用新规则集。

Console

删除规则集

1. 打开 GameLift 控制台位于https://console.aws.amazon.com/gamelift/.
2. 切换到您在中创建规则集的区域。
3. 在导航窗格中，选择FlexMatch、对战规则集.
4. 在存储库的对战规则集页面上，选择要删除的规则集，然后选择Delete.
5. 在删除规则集对话框中，选择Delete以确认删除。

   Note

   如果配对配置正在使用规则集， GameLift显示错误消息 (无法删除规则集)。如果发生这种情况，更改对战配置以使用其他规则集，然后再试一次。要了解一个规则集目前正被哪些对战配置使用，请选择该规则集名称，查看其详细信息页面。

Amazon CLI

删除规则集

打开命令行窗口，然后使用命令delete-matchmaking-rule-set删除对战规则集。

如果配对配置正在使用规则集， GameLift 返回错误消息。如果发生这种情况，更改对战配置以使用其他规则集，然后再试一次。要获取使用规则集的对战配置使用规则集，请使用命令describe-matchmaking-configurations并指定规则集名称。

此示例命令检查对战规则集的使用情况，然后删除该规则集。

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

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

FlexMatch 规则集示例

FlexMatch 规则集可以涵盖各种对战情况。以下示例符合 FlexMatch 配置结构和属性表达式语言。完整复制这些规则集或根据需要选择组件。

有关使用 FlexMatch 规则和规则集的更多信息，请参阅以下主题：

- 构建 FlexMatch 规则集 (p. 12)
- 设计 FlexMatch 规则集 (p. 13)
- FlexMatch 规则集架构 (p. 49)
- FlexMatch 规则语言 (p. 49)

Note

当评估包含多个玩家的对战票证时，请求中的所有玩家都必须满足对战要求。

示例 1：使用均匀匹配的玩家创建两个队

此示例说明如何按照以下说明设置两个势均力敌的玩家对战团队。

- 创建两个玩家团队。

    - 每个团队包含四到八个玩家。
    - 最终团队必须具有相同数量的玩家。
- 附上玩家的技能水平 (如果未提供，则默认为 10)。
- 选择与其他玩家技能水平相当的玩家。确保这两个团队的平均玩家技能在 10 点以内。
- 如果未能快速填充对战游戏，则放宽玩家技能要求以在合理的时间内完成对战游戏。
    - 5 秒之后，扩大搜索范围以允许创建平均玩家技能在 50 点以内的团队。
    - 15 秒之后，扩大搜索范围以允许创建平均玩家技能在 100 点以内的团队。

使用此规则集的说明：

- 此示例允许创建包含 4 到 8 位玩家的任何规模的团队 (但两个团队的规模必须相同)。对于在有效规模范围内的团队，对战构建器会尽量尝试匹配允许的最大玩家数量。
- `FairTeamSkill` 规则集可确保根据玩家技能构建对战团队。要对每个新的潜在玩家评估此规则，FlexMatch 会暂时将玩家加入团队并计算平均值。如果规则失败，则不会将潜在玩家添加到对战游戏。
- 由于这两个团队具有相同的结构，您可以选择仅创建一个团队定义，并将团队数量设置为“2”。在这种情况下，如果您将团队命名为“aliens”，那么为您的团队分配的名称将为“aliens_1”和“aliens_2”。

```
{
    "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：创建参差不对等的团队 (猎人对战怪物)

此示例描述了一组玩家搜捕单个怪物的游戏模式。玩家可以选择猎人或怪物角色。猎人指定他们希望对战的怪物的最低技能水平。可随时间推移放宽猎人团队的最小规模以完成对战游戏。此方案规定了以下说明：

- 创建一个包含五个猎人的团队。
- 创建一个包含单个怪物的团队。
- 包含以下玩家属性：
  - 玩家的技能水平 (如果未提供，则默认为 10)。
  - 玩家首选的怪物技能水平 (如果未提供，则默认为 10)。
  - 玩家是否想成为怪物 (如果未提供，则默认为 0 或 false)。
- 根据以下条件选择将成为怪物的玩家：
  - 玩家必须请求怪物角色。
  - 玩家必须满足或超过已添加到猎人团队的玩家的首选最高技能水平。
- 根据以下条件选择将加入猎人团队的玩家：
  - 请求怪物角色的玩家无法加入猎人团队。
  - 如果怪物角色已填充，玩家必须要有低于建议怪物技能的怪物技能水平。
- 如果对战游戏未快速填满，则放宽猎人团队的最小规模，如下所示：
  - 30 秒后，允许启动猎人团队只包含四个玩家的游戏。
  - 60 秒后，允许启动猎人团队只包含三个玩家的游戏。

使用此规则集的说明：

- 通过为猎人和怪物分别创建两个单独的团队，您可以根据不同的条件评估团队成员。

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

示例 3：设置团队级要求和延迟限制

此示例说明如何设置玩家团队，并为每个团队应用规则集，而不是为每个玩家应用规则集。它使用单个定义创建三个平均的对战团队。它还会设置所有玩家的最大延迟。随着时间的推移可以放宽延迟最大值以完成对战游戏。此方案规定了以下说明：

- 创建三个玩家团队。
  - 每个团队包含三到五个玩家。
  - 最终团队必须包含数量相同或基本相同的玩家 (一个团队中)。
- 包含以下玩家属性：
  - 玩家的技能水平 (如果未提供，则默认为 10)。
  - 玩家的人物角色 (如果未提供，则默认为“peasant”)。
- 选择与对战游戏中其他玩家技能水平相当的玩家。
  - 确保每个团队的平均玩家技能在 10 点以内。
- 将团队数量限制为“medic”角色的以下数量：
  - 整个对战游戏最多可以包含 5 个医生。
- 仅匹配报告 50 毫秒或更少延迟的玩家。
- 如果对战游戏未快速填满，则放宽玩家的延迟要求，如下所示：
  - 10 秒之后，允许将玩家的延迟时间值增加到 100 毫秒。

- 20 秒之后，允许将玩家的延迟时间值增加到 150 毫秒。

使用此规则集的说明：

- 规则集可确保根据玩家技能构建对战团队。评估FairTeamSkill规则，FlexMatch 会暂时将潜在玩家加入团队并计算团队中玩家的平均技能。然后，它将规则与两个团队中的玩家平均技能进行比较。如果规则失败，则不会将潜在玩家添加到对战游戏。
- 团队和对战游戏级别的要求 (医生总数) 通过一组规则实现。此规则类型会获取所有玩家的角色属性列表，并针对最大数量进行检查。使用 `flatten` 为所有团队中的所有玩家创建列表。
- 根据延迟进行评估时，请注意以下几点：
  - 延迟数据在对战请求中作为玩家对象的一部分提供。它不是玩家属性，因此无需列出。
  - 对战构建器按区域评估延迟。延迟高于最大延迟的任何区域都将被忽略。要让对战游戏接受，玩家必须至少有一个区域的延迟低于最大延迟。
  - 如果对战请求忽略一个或多个玩家的延迟数据，则会拒绝所有对战游戏的请求。

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

示例 4：使用显式排序查找最佳匹配

此示例设置了包含三个玩家的两个团队的简单对战游戏。它介绍了如何使用显式排序规则才能尽快找到最佳对战游戏。这些规则会对所有活动对战票证进行排序，以根据某些关键要求创建最佳对战游戏。此方案根据以下说明实现：

- 创建两个玩家团队。
- 每个团队只包含三个玩家。
- 包含以下玩家属性：
  - 经验等级 (如果未提供，则默认为 50)。
  - 首选游戏模式 (可以列出多个值) (如果未提供，则默认为“coop”和“deathmatch”)。
  - 首选游戏地图，包括地图名称和首选权重 (如果未提供，则默认使用 `"defaultMap"` 和权重 100)。
- 设置预排序：
  - 根据与基准点玩家相同的游戏地图的首选项来对玩家排序。玩家可以有多个最喜欢的游戏地图，因此本示例使用首选项值。
  - 根据玩家与基准点玩家的经验等级的接近程度对玩家排序。采用这种排序方式，所有玩家在所有团队中的经验等级会尽可能地接近。
- 所有团队中的所有玩家必须至少选择一个共同的游戏模式。
- 所有团队中的所有玩家必须至少选择一个共同的游戏地图。

使用此规则集的说明：

- 游戏地图排序采用与 mapPreference 属性值进行比较的绝对排序。由于它是规则集中的第一个，此排序将首先执行。
- 经验排序采用比较潜在玩家的技能级别与基准点玩家的技能的距离排序。
- 排序按它们在规则集中的排列顺序执行。在这种情况下，玩家先按游戏地图首选项进行排序，然后按经验等级排序。

```
{
    "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
    }]
```

```
}
```

示例 5：查找跨多个玩家属性的交集

此示例说明如何使用集合规则，在两个或更多玩家属性中查找交集。在处理集合时，您可以对单个属性使用 `intersection` 操作，并对多个属性使用 `reference_intersection_count` 操作。

为了说明这种方法，此示例会在对战游戏中根据玩家的角色首选项来评估玩家。该示例游戏是一个“混战”风格的游戏，对战游戏中的所有玩家都是对手。每个玩家需要 (1) 为自己选择一个角色，(2) 选择他们希望作为对手的角色。我们需要一个规则，该规则可确保对战游戏中的每位玩家所使用的角色都位于所有其他玩家的首选对手列表中。

该示例规则集描述了一个具有以下特征的对战游戏：

- 团队结构：一个团队 5 个玩家
- 玩家属性：
  - myCharacter：玩家的选定角色。
  - preferredOpponents：玩家想对战的角色列表。
- 对战规则：如果每个正在使用的角色都位于每个玩家的首选对手列表中，则潜在的对战游戏是可

为了实施该对战游戏规则，此示例使用具有以下属性值的集合规则：

- 操作 — 用途`reference_intersection_count`操作来评估测量值中的每个字符串列表与参考值中的字符串列表的相交情况。
- 测量 — 使用`flatten`属性表达式来创建一个字符串列表的列表，其中每个字符串列表包含一个玩家的myCharacter属性值。
- 参考值 — 使用`set_intersection`属性表达式来创建所有内容的字符串列表preferredOpponents对战游戏中的每个玩家通用的属性值。
- 限制 —`minCount`设置为 1，以确保每个玩家的选定角色（测量值中的一个字符串列表）匹配至少一个对所有玩家通用的首选对手。 (参考值中的字符串)。
- 扩展 — 如果对战游戏在 15 秒内未满员，则放松最小交集要求。

此规则的流程如下：

1. 将一位玩家添加到潜在对战游戏。重新计算参考值 (字符串列表)，以便包含与新玩家的首选对手列表的交集。重新计算测量值 (字符串列表的列表)，以便将新玩家的选定角色作为新的字符串列表添加。
2. Amazon GameLift 会验证测量值（玩家的选定角色）中的每个字符串列表与参考值（玩家的首选对手）中的至少一个字符串相交。在本示例中，由于测量值中的每个字符串列表中仅包含一个值，因此交集为 0 或 1。
3. 如果测量值中的任何字符串列表与参考值字符串列表不相交，则该规则失败，并且从潜在对战游戏中删除该新玩家。
4. 如果对战游戏在 15 秒内未满员，则放弃对手匹配要求，以便在对战游戏中填满剩余的玩家位置。

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

示例 6：比较所有玩家的属性

此示例说明如何在一组玩家之间比较玩家属性。

该示例规则集描述了一个具有以下特征的对战游戏：

- 团队结构：两个单人游戏团队
- 玩家属性：
  - gameMode：玩家选择的游戏类型（如果未提供，则默认设置为 “回合制”）。
  - gameMap：游戏世界由玩家选择（如果未提供，则默认设置为 1）。
  - 字符：玩家选择的角色（无默认值，表示玩家必须指定角色）。
- 对战规则：匹配的玩家必须满足以下要求：
  - 玩家必须选择相同的游戏模式。
  - 玩家必须选择相同的游戏地图。
  - 玩家必须选择不同的角色。

使用此规则集的说明：

- 为了实施该对战游戏规则，此示例使用比较规则来检查所有玩家的属性值。对于游戏模式和地图，该规则验证值是相同的。对于角色，该规则验证值是不同的。
- 此示例使用一个玩家定义与数量属性来创建两个玩家团队。为团队分配了下列名称：“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])"]
    }]
}
```

示例 7：创建大型比赛

此示例说明如何为超过 40 位玩家的对战设置规则集。当规则集描述 maxPlayer 总数大于 40 的团队时，该团队会被处理为大型对战。在 设计 FlexMatch 大型匹配规则集 (p. 17) 中了解更多信息。

此示例规则集使用以下说明创建对战：

- 创建一个最多有 200 位玩家的团队，玩家人数最低要求为 175 位。
- 平衡标准：根据相似的技能水平选择玩家。所有玩家均必须报告其技能级别才能被匹配。
- 批处理首选项：创建对战时按类似的平衡条件对战游戏进行分
- 延迟规则：将可接受的最大玩家延迟设置为 150 毫秒。
- 如果未能快速填充对战游戏，则放宽要求以在合理的时间内完成匹配。
  - 10 秒后，接受有 150 位玩家的团队。
  - 12 秒后，将可接受的最大延迟提高到 200 毫秒。
  - 15 秒后，接受有 100 位玩家的团队。

使用此规则集的说明：

- 由于算法使用“最大群体”批处理首选项，玩家会根据平衡条件排序。因此，对战往往填得更满，包含技能更加类似的玩家。所有玩家均符合可接受的延迟要求，但可能无法在他们的位置获得可能的最佳延迟。

- 此规则集中使用的算法策略“最大群体”是默认设置。要使用默认设置，您可以选择忽略此设置。
- 如果您已启用了对战回填，请不要太快地放宽玩家计数要求，否则，您最后可能会有太多部分填充的游戏会话。在 放松大型比赛要求 (p. 19) 中了解更多信息。

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

示例 8：创建多队大型比赛

此示例说明如何为有多个团队可能超过 40 位玩家的对战设置规则集。此示例说明如何使用一个定义创建多个相同的团队，以及规模不对称的团队如何在对战创建过程中填充。

此示例规则集使用以下说明创建对战：

- 创建十个相同的“猎人”团队，每个团队最多有 15 位玩家，并创建一个刚好有 5 位玩家的“怪物”团队。
- 平衡标准：根据怪物杀人数选择玩家。如果玩家未报告其技能计数，请使用默认值 5。
- 批处理首选项：根据玩家报告最快玩家延迟的区域对玩家进行分组。
- 延迟规则：将可接受的最大玩家延迟设置为 200 毫秒。
- 如果未能快速填充对战游戏，则放宽要求以在合理的时间内完成匹配。
  - 15 秒后，接受有 10 位玩家的团队。
  - 20 秒后，接受有 8 位玩家的团队。

使用此规则集的说明：

- 此规则集定义最多可能容纳 155 位玩家的团队，这使其成为大型对战。（10 x 15 个猎人 + 5 个怪物 = 155 个）
- 由于算法使用“最快区域”批处理首选项，玩家更可能被置于他们报告更快延迟的区域，而不是他们报告较高（但可接受）延迟的区域。同时，可能有更少玩家的对战和平衡条件（怪物技能的数量）可能会变化更大。
- 当为多团队定义（数量 > 1）定义了扩展时，扩展将应用于使用该定义创建的所有团队。因此，通过放宽猎人团队的最低玩家设置，全部十个猎人团队都会受到均等的影响。
- 由于此规则集经过优化以最大程度地减少玩家延迟，延迟规则将充当“捕获全部”方法来排除没有可接受连接选项的玩家。我们不需要放宽此要求。
- 以下是 FlexMatch 在任何扩展生效之前填充此规则集的匹配情况：
  - 还没有团队达到 minPlayers 计数。猎人团队有 15 个空闲位置，怪物团队有 5 个空闲位置。
    - 前 100 个玩家被分配（每个团队 10 个）到十个猎人团队。
    - 接下来 22 个玩家按顺序分配（每个团队 2 个）到猎人团队和怪物团队。
  - 猎人团队已达到每个团队 12 个玩家的 minPlayers 计数。怪物团队有 2 个玩家，还未达到 minPlayers 计数。
    - 接下来的三个玩家被分配到怪物团队。
  - 所有团队均达到 minPlayers 计数。每个猎人团队都有三个空闲位置。怪物团队已满。
    - 最后 30 个玩家按顺序分配到猎人团队，确保所有猎人团队具有几乎相同的规模（加上或减去一个玩家）。
- 如果您已为使用此规则集创建的对战启用了回填，请不要太快地放宽玩家计数要求，否则，您最后可能会有太多部分填充的游戏会话。在 放松大型比赛要求 (p. 19) 中了解更多信息。

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

示例 9：与具有相似属性的玩家创建大型比赛

此示例说明如何为与两个团队的对战设置规则集batchDistance. 在示例中：

- 这些区域有：SimilarLeague规则确保对战游戏中的所有玩家都有league其他玩家中有 2 个以内。
- 这些区域有：SimilarSkill规则确保对战游戏中的所有玩家都有skill其他玩家的 10 个以内。如果玩家已经等了 10 秒钟，则距离将扩大到 20 秒。如果玩家一直在等待 20 秒，则距离将扩大到 40 秒。
- 这些区域有：SameMap规则确保对战游戏的所有玩家都要求相同的map.
- 这些区域有：SameMode规则确保对战游戏的所有玩家都要求相同的mode.

```
{
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "name": "red",
        "minPlayers": 100,
        "maxPlayers": 100
    }, {
        "name": "blue",
        "minPlayers": 100,
        "maxPlayers": 100
    }],
    "algorithm": {
        "strategy":"balanced",
        "balancedAttribute": "skill",
        "batchingPreference":"fastestRegion"
    },
    "playerAttributes": [{
        "name": "league",
        "type": "number"
    },{
        "name": "skill",
        "type": "number"
    },{
        "name": "map",
        "type": "string"
    },{
        "name": "mode",
        "type": "string"
    }],
    "rules": [{
        "name": "SimilarLeague",
        "type": "batchDistance",
        "batchAttribute": "league",
        "maxDistance": 2
    }, {
        "name": "SimilarSkill",
        "type": "batchDistance",
        "batchAttribute": "skill",
        "maxDistance": 10
    }, {
        "name": "SameMap",
        "type": "batchDistance",
        "batchAttribute": "map"
```

```
    }, {
        "name": "SameMode",
        "type": "batchDistance",
        "batchAttribute": "mode"
    }],
    "expansions": [{
        "target": "rules[SimilarSkill].maxDistance",
        "steps": [{
            "waitTimeSeconds": 10,
            "value": 20
        }, {
            "waitTimeSeconds": 20,
            "value": 40
        }]
    }]
}
```

示例 10：使用复合规则与具有相似属性或类似选择的玩家创建匹配

此示例说明如何为与两个团队的对战设置规则集compound. 在示例中：

- 这些区域有：SimilarLeagueDistance规则确保对战游戏中的所有玩家都有league其他玩家中有 2 个以内。
- 这些区域有：SimilarSkillDistance规则确保对战游戏中的所有玩家都有skill其他玩家的 10 个以内。如果玩家已经等了 10 秒钟，则距离将扩大到 20 秒。如果玩家一直在等待 20 秒，则距离将扩大到 40 秒。
- 这些区域有：SameMapComparison规则确保对战游戏的所有玩家都要求相同的map.
- 这些区域有：SameModeComparison规则确保对战游戏的所有玩家都要求相同的mode.
- 这些区域有：CompoundRuleMatchmaker如果至少满足以下任意条件，则确保匹配项：
  - 比赛中的玩家也要求同样map而且相同的mode.
  - 比赛中的玩家具有可比性skill和league属性。

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

设置 FlexMatch 事件通知

如果您使用的是 GameLift FlexMatch 对战，您需要一种方法来跟踪单独对战请求的状态。实施事件通知是跟踪对战事件的快速有效的方法。所有投入实际生产的游戏，或具有大量对战活动的预生产中的游戏都应使用事件通知。

有两个选项可用于设置事件通知。你可以使用亚马逊 CloudWatch Events，其中包括一套工具，可用于管理事件和对其采取行动。要么, 您可以设置自己的Amazon Simple Notification Service (Amazon SNS) 主题并配置您的对战构建器，让对战事件通知直接发送到主题。

要查看 FlexMatch 那些事件 GameLift 发射，请参阅FlexMatch 对战事件 (p. 61).

设置 CloudWatch 事件

GameLift 自动将所有对战事件发送到 CloudWatch 事件。与 CloudWatch 大事记，您可以设置规则，让对战事件路由到一系列目标以进行处理，包括 SNS 主题和其他Amazon处理服务。例如，您可以设置一个规

则，以路由事件"PotentialMatchCreated"到Amazon Lambda处理玩家接受的函数。有关如何使用的更多信息 CloudWatch 事件，包括一系列教程，请参阅Amazon 入门 CloudWatch 事件中的亚马逊 CloudWatch Events 用户指南.

如果您计划使用 CloudWatch 大事记，配置您的对战构建器时，可以将通知目标字段留空；如果您希望使用两个选项，则参考 SNS 主题。

您可以访问 GameLift 对战 CloudWatch 中的事件CloudWatch 控制台. 有关更多信息，请参阅 。登录 Amazon CloudWatch 控制台. CloudWatch 事件标识每个对接会服务事件 (GameLift), 对战名称和对战票证书。

设置 Amazon SNS 主题

您可以有 GameLift 发布所有事件 FlexMatch 对战生成Amazon SNSNS主题。

创建 SNS 主题 GameLift 事件通知

1. 打开 Amazon SNS 控制台。
2. 在导航窗格中，选择 Topics (主题)。
3. 在 Topics（主页）页面上，选择 Create topic（创建主题）。
4. 在 控制台中，创建一个主题。有关更多信息，请参阅 。使用 创建主题Amazon Web Services Management Console中的Amazon Simple Notificat.
5. 在存储库的详细信息页面，选择编辑.
6. 在存储库的编辑您的主题页面，展开访问策略-可选的，然后从下面添加粗体语法Amazon Identity and Access Management(IAM) 策略声明到现有策略的结尾。(为清晰起见此处显示了整个策略。) 确保将 Amazon Resource Name (ARN) 详细信息用于您自己的 SNS 主题和 GameLift 对战。

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

7. 选择 Save changes（保存更改）。

配置主题订阅以调用 Lambda 函数

您可以使用发布到 Amazon SNS 主题的事件通知调用 Lambda 函数。配置对战构建器时，请务必将通知目标设置为 SNS 主题的 ARN。

以下Amazon CloudFormation模板配置对名为的 SNS 主题的订阅`MyFlexMatchEventTopic`调用 Lambda 函数`FlexMatchEventHandlerLambdaFunction`. 该模板创建了一个 IAM 权限策略，该策略允许 GameLift 写到 SNS 主题。最后，它为 SNS 主题添加 Lambda 函数调用 Lambda 函数的权限。

```
FlexMatchEventTopic:
  Type: "AWS::SNS::Topic"
  Properties:
    KmsMasterKeyId: alias/aws/sns #Enables server-side encryption on the topic using an
 Amazon 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
```

为 FlexMatch 准备游戏

使用 GameLift FlexMatch 为游戏添加对战玩家。FlexMatch 可与托管 GameLift 解决方案一起用于自定义游戏服务器和实时服务器。

FlexMatch 可将对战服务与自定义规则引擎搭配使用。这样，您就可以设计如何根据适用于您的游戏的玩家属性和游戏模式匹配对战玩家，然后依靠 FlexMatch 来管理构成玩家组的基本要素，并将它们放置到游戏中。请参阅FlexMatch 规则集示例 (p. 21)中有关自定义对战的更多详细信息。

FlexMatch 基于队列功能进行构建。对战游戏组成之后，FlexMatch 将对战游戏的详细信息处理成供您选择的队列。此队列可在您的 Amazon GameLift 队组中搜索可用的托管资源，并为对战游戏启动新的游戏会话。

此部分的主题介绍如何向您的游戏服务器和游戏客户端添加对战支持。要为游戏创建对战构建器，请参阅 正在建一个 GameLift FlexMatch 媒人 (p. 8)。有关 FlexMatch 工作原理的更多信息，请参阅。Amazon GameLift FlexMatch 的工作原理 (p. 2).

Add FlexMatch 到游戏客户端

本主题介绍如何添加 FlexMatch 为您的客户端游戏服务提供配对支持. 无论您使用什么，该过程基本上是相同的 FlexMatch 和 GameLift 托管主机或其他托管解决方案。了解相关更多信息 FlexMatch 以及如何为游戏设置自定义对战构建器，请参阅以下主题：

- FlexMatch 与 GameLift 托管集成 (p. 7)
- Amazon GameLift FlexMatch 的工作原理 (p. 2)
- 正在建一个 GameLift FlexMatch 媒人 (p. 8)
- FlexMatch 规则集示例 (p. 21)

启用 FlexMatch 在对战游戏中，添加以下功能：

- 准备为一个或多个玩家请求对战（必需）。
- 跟踪对战请求的状态（必需）。
- 要求玩家接受建议的对战游戏（可选）。
- 在为新的对战创建游戏会话之后，获取玩家连接信息并加入游戏。

准备为玩家申请配对

我们强烈建议您的游戏客户端通过客户端游戏服务提出对战请求。通过使用可信源，您可以更轻松地防范黑客攻击和虚假玩家数据。如果您的游戏具有会话目录服务，那么这是一个用于处理对战请求的好选项。

要准备您的客户端服务，请执行以下任务：

- 添加 GameLift API。您的客户服务使用中的功能 GameLift API，它是AmazonSDK。请参阅GameLift 用于客户端服务的 SDK了解有关的更多信息AmazonSDK 并下载最新版本。将此 SDK 添加到您的客户端服务项目。
- 设置对战票证系统。必须向所有对战请求分配一个唯一票证 ID。您需要一个生成唯一 ID 并将其分配给新对战请求的机制。票证 ID 可以使用任意字符串格式，最多 128 个字符。
- 获取对战构建器信息。获取您计划要使用的对战配置的名称。您还需要对战构建器的必需玩家属性列表，这在对战构建器规则集中定义。

- 获取玩家数据。设置获取各个玩家相关数据的方法。这包括玩家 ID、玩家属性值，以及玩家可能接入游戏的各个区域的更新延迟数据。
- （可选）启用对战回填。确定您想要如何回填现有的匹配游戏。如果您的对战构建器将回填模式设置为“手动”，您可能需要为您的游戏添加回填支持。如果回填模式设置为“自动”，您可能需要一种为单个游戏会话关闭此模式的方法。在 使用 FlexMatch 回填现有游戏 (p. 43) 中了解有关管理对战回填的更多信息。

请求玩家的对战

将代码添加到您的客户端服务来创建和管理 FlexMatch 对战构建器的对战请求。请求的过程 FlexMatch 对于使用配对的游戏，配对是相同的 FlexMatch 和 GameLiftmanaged 托管和使用以下内容的游戏 FlexMatch 作为独立的解决方案。

创建对战请求：

- 调用 GameLift APIStartMatchmaking. 每个请求都必须包含以下信息。

  对战构建器

  要用于请求的对战配置的名称。 FlexMatch 将各个请求放置到指定对战构建器的池中，并将根据对战构建器的配置方式来处理请求。这包括强制施加时间限制，是否请求玩家接受匹配，在放置生成的游戏会话时使用哪个队列，等等。在 设计一个 FlexMatch 媒人 (p. 8) 中了解有关对战构建器和规则集的更多信息。

  票证 ID

  分配给请求的唯一票证 ID。与请求相关的所有信息（包括事件和通知）都将引用票证 ID。

  玩家数据

  您要为其创建对战的玩家的列表。如果根据对战规则和延迟最低值，请求中的任意玩家不满足对战要求，则对战请求绝不会生成成功的对战。您可在一个对战请求中包括最多十位玩家。当请求中有多个玩家时， FlexMatch 尝试创建单个对战并将所有玩家分配到相同团队中（随机选择）。如果请求包含的玩家数太多，无法放在一个对战团队中，则对战请求失败。例如，如果您设置了对战构建器，以创建 2 对 2 对战（两个团队，每个团队两个玩家），您无法发送包含两个以上玩家的对战请求。

  Note

  一个玩家 (通过其玩家 ID 标识) 一次只能包括在一个有效对战请求中。在您为玩家创建新请求时，将自动取消任何具有相同玩家 ID 的有效对战票证。

  对于每个列出的玩家，请提供以下数据：

  - 玩家 ID— 每个玩家必须具有一个唯一的玩家 ID，该 ID 由您生成。请参阅生成玩家 ID.
  - 玩家属性— 如果所使用的对战构建器需要玩家属性，请求必须为每个玩家提供这些属性。必需的玩家属性在对战构建器的规则集中定义，同时还会指定属性的数据类型。玩家属性仅在规则集指定属性的默认值时是可选的。如果对战请求未提供所有玩家的必需玩家属性，对战请求可能永远无法成功。在构建 FlexMatch 规则集 (p. 12)和FlexMatch 规则集示例 (p. 21)中了解有关对战构建器规则集和玩家属性的更多信息。
  - 玩家延迟— 如果所使用的对战构建器有玩家延迟规则，请求必须报告每个玩家的延迟。玩家延迟数据是显示每个玩家的一个或多个值的列表。它表示对战构建器的队列中各个区域的玩家体验的延迟。如果请求中未包含延迟值，玩家将无法匹配，请求将失败。

检索对战请求详细信息：

- 发送对战请求后，您可以通过调用DescribeMatchmaking使用请求的票证 ID。此调用将返回请求信息，包括当前状态。成功完成请求之后，票证还将包含游戏客户端连接到对战所需的信息。

取消对战请求：

- 您可以随时通过调用，取消对战请求StopMatchmaking使用请求的票证 ID。

跟踪对战事件

设置通知，以跟踪事件 GameLift 用于配对过程的发射. 您可以直接设置通知，也可以通过创建 SNS 主题或使用亚马逊来设置通知 EventBridge. 有关设置通知的更多信息，请参阅设置 FlexMatch 事件通知 (p. 35)。设置通知之后，请在客户端服务上添加侦听器以检测事件并根据需要做出响应。

在经过相当长一段时间而未通知的情况下，最好定期轮询状态更新来作为通知的备用手段。为了最大限度地减少对对战性能的影响，请务必在提交对战票证后或最后一次收到通知后，等待至少 30 秒后再轮询。

通过调用，检索对战请求票证，包括当前状态DescribeMatchmaking使用请求的票证 ID。我们建议轮询频率不要超过每 10 秒一次。此方法仅在低容量开发场景中使用。

> Note
>
> 在使用大量对战场景之前，您应该使用事件通知设置游戏，例如进行预生产负载测试。公开发布版中的所有游戏都应该使用通知，而不考虑容量。连续轮询方法仅适用于对战使用率较低的开发中的游戏。

要求玩家接受

如果您使用的是开启了玩家接受的对战构建器，请将代码添加到您的客户端服务来管理玩家接受过程。对于使用的游戏，管理玩家接受程度的过程是相同的 FlexMatch 和 GameLift-托管托管和使用托管的游戏 FlexMatch 作为独立的解决方案。

要求玩家接受建议的对战游戏：

1. 检测建议的对战游戏何时需要玩家接受。监控对战票证以检测状态更改为 `REQUIRES_ACCEPTANCE` 的情况。更改此状态会触发 FlexMatch 事件`MatchmakingRequiresAcceptance`.
2. 从所有玩家获取接受信息。创建一个机制，以在对战票证中向每个玩家呈现建议的对战游戏详细信息。玩家必须能够表明他们接受或拒绝建议的对战游戏。您可以通过调用来检索对战游戏详细信息DescribeMatchmaking. 在对战构建器撤消建议的对战游戏之前，玩家仅有有限的响应时间。
3. 向报告玩家响应 FlexMatch. 通过调用，报告玩家响应AcceptMatch要么接受，要么拒绝。对战请求中的所有玩家必须接受对战游戏才能继续。
4. 处理具有失败接受的票证。当建议的对战游戏中的任何一个玩家拒绝对战游戏，或者未能在接受时限内响应时，请求失败。接受比赛的玩家的门票将自动返回到彩票池。未接受对战的玩家的票证将变为失败状态，并且不会再进行处理。对于多位玩家的彩票，如果彩票中的任何玩家不接受比赛，则整张彩票失效。

Connect 对战游戏

将代码添加到客户端服务以处理已成功形成的对战（状态）`COMPLETED`或`MatchmakingSucceeded`)。这包括向游戏客户端通知对战游戏的玩家和传递连接信息。

适用于使用以下功能的游戏 GameLift 托管托管，当对战请求成功完成时，游戏会话连接信息将添加到对战票证。通过调用，检索已完成的对战票证DescribeMatchmaking. 连接信息包括游戏会话的 IP 地址和端口，以及每个玩家 ID 的玩家会话 ID。在 GameSessionConnectionInfo 中了解更多信息。您的游戏客户端可以使用此信息直接连接到对战游戏会话。连接请求应该包含玩家会话 ID 和玩家 ID。此数据将连接的玩家与游戏会话的对战数据相关联，其中包括团队分配（请参阅GameSession)。

适用于使用其他托管解决方案的游戏，包括 GameLift FleetIQ，你必须建立一种机制，使比赛玩家能够连接到相应的游戏会话。

对战请求示例

以下代码段为多个不同的对战构建器生成对战请求。如文中所述，请求必须提供所使用的对战构建器需要的玩家属性（在对战构建器的规则集中定义）。提供的属性必须使用在规则集中定义的相同的数据类型：数字 (N) 或字符串 (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)
```

将 FlexMatch 添加到 GameLift 托管的游戏服务器

本主题介绍如何将 FlexMatch 对战支持添加到使用 GameLift 托管主机的自定义游戏服务器。要了解有关向游戏添加 FlexMatch 的更多信息，请参阅以下主题：

- Amazon GameLift FlexMatch 的工作原理 (p. 2)
- FlexMatch 与 GameLift 托管集成 (p. 7)

本主题中的信息假定您已将 GameLift Server SDK 成功集成到您的游戏服务器项目中，如中所述。将 GameLift 添加到您的游戏服务器. 完成此工作后，您便有了所需的大部分机制。本主题中的各个部分介绍了处理使用 FlexMatch 设置的游戏所需的其他工作。

设置您的游戏服务器以进行对战

要设置您的游戏服务器来处理已对战的游戏，请完成以下任务。

1. 启动使用对战创建的游戏会话。要申请新的游戏会话，GameLift 发送了`onStartGameSession()`请求使用游戏会话对象（请参阅GameSession)。您的游戏服务器使用游戏会话信息 (包括自定义的游戏数据) 以启动请求的游戏会话。有关更多详细信息，请参阅启动游戏会话.

   对于已匹配的游戏，游戏会话对象还包含一组对战构建器数据。对战构建器数据包含您的游戏服务器启动新的对战游戏会话所需的信息。这包括对战的团队结构、团队分配以及可能与您的游戏相关的某些玩家属性。例如，您的游戏可以根据平均玩家技能级别解锁某些功能，或根据玩家的首选项选择地图。在使用对战构建器数据 (p. 42) 中了解更多信息。
2. 处理玩家连接。当连接到已匹配的游戏时，游戏客户端将引用玩家 ID 和玩家会话 ID（请参阅验证新玩家)。您的游戏服务器使用玩家 ID 将传入玩家与对战构建器数据中的玩家信息进行关联。对战构建器数据将标识玩家的团队分配，并且可能提供在游戏中正确地代表玩家的其他信息。
3. 在玩家离开游戏时进行报告。确保您的游戏服务器在调用服务器 API`RemovePlayerSession()`报告掉落的玩家（请参阅报告玩家会话结束)。如果您使用 FlexMatch 回填填填填现有游戏中的空位置，此步骤非常重要。如果您的游戏通过客户端游戏服务发起回填请求，此步骤非常重要。在中了解有关实施 FlexMatch 回填的更多信使用 FlexMatch 回填现有游戏 (p. 43).
4. 为现有的已对战游戏会话请求新玩家（可选）。确定您想要如何回填现有的匹配游戏。如果您的对战构建器将回填模式设置为“手动”，您可能需要为您的游戏添加回填支持。如果回填模式设置为“自动”，您可能需要一种为单个游戏会话关闭此模式的方法。例如，您可能想要在到达游戏中的某个点后停止回填游戏会话。在 使用 FlexMatch 回填现有游戏 (p. 43) 中了解有关管理对战回填的更多信息。

使用对战构建器数据

您的游戏服务器必须能够识别和使用 GameSession 对象中的游戏信息。每当启动或更新游戏会话时，GameLift 服务就会将这些对象传递到您的游戏服务器。核心游戏会话信息包括游戏会话 ID 和名称、最大玩家数量、连接信息和自定义游戏数据 (如果提供)。

对于使用 FlexMatch 创建的游戏会话，`GameSession`对象还包含一组对战构建器数据。除了唯一的对战 ID，它将标识创建对战的对战构建器并描述团队、团队分配和玩家。它包括原始对战请求中的玩家属性 (请参阅 Player 对象)。它不包括玩家延迟；如果您需要当前玩家的延迟数据 (如对战回填)，建议您获取最新数据。

> Note
>
> 对战构建器数据指定完整的对战配置 ARN，该 ARN 将标识配置名称，Amazon账户和地区。在从游戏客户端或服务请求对战回填时，仅需要配置名称。您可以通过解析出“:matchmakingconfiguration/”后面的字符串来提取配置名称。在显示的示例中，对战配置名称为“MyMatchmakerConfig”。

以下 JSON 显示一组典型的对战构建器数据。此示例描述了一个两人游戏，该游戏根据技能评级和获得的最高级别匹配玩家。对战构建器还根据角色进行对战，并且确保对战的玩家至少具有一个共同的地图首选项。在这种情况下，游戏服务器应该能够确定哪个地图最受偏爱，并在游戏会话中使用它。

```
{
 "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 FlexMatch 回填现有游戏

对战回填使用 FlexMatch 机制为现有的已匹配游戏会话寻找新玩家。尽管你总是可以将玩家添加到任何游戏中（请参阅将玩家接入游戏会话），对战回填可确保新玩家满足与当前玩家相同的对战条件。此外，对战回填会将新玩家分配到团队，管理玩家接受，并将更新的对战信息发送到游戏服务器。在FlexMatch 对战过程 (p. 3)中了解有关对战回填的更多信息。

> **Note**
>
> FlexMatch 回填当前不能使用用于使用实时服务器的游戏。

回填机制有两种类型：

- 要填充以少于允许的最大玩家数开始的游戏会话，请启用自动回填。
- 要取代正在进行的游戏会话中退出的玩家，请向游戏服务器添加功能以发送回填请求。

启用自动回填

使用自动对战回填时，GameLift 会话从一个或多个玩家槽位开始时，都会自动触发回填请求。此功能允许游戏在找到最少匹配玩家数量后立即开始，并在匹配到其他玩家后填充剩余槽位。您可以随时选择停止自动回填。

例如，如果您有一个可容纳六到十名玩家的游戏。FlexMatch 最初定位六名玩家，组建对战并开始新的游戏会话。使用自动回填时，新游戏会话可以立即要求增加四名玩家。根据游戏风格，我们可能希望允许新玩家在游戏会话期间随时加入。或者，我们可能希望在初始设置阶段之后、游戏开始之前停止自动回填。

要向您的游戏添加自动回填，请对您的游戏进行以下更新。

1. 启用自动回填。自动回填在对战配置中管理。启用后，它将用于与该媒人创建的所有匹配游戏会话一起使用。当游戏会话在游戏服务器上启动时，GameLift 便会开始为未满游戏会话生成回填请求。

   要打开自动回填，请打开对战配置并将回填模式设置为“AUTOMATIC”(自动)。有关更多详细信息，请参阅 创建对战配置 (p. 9)
2. 启用回填优先级. 自定义匹配流程，以便在创建新匹配之前优先填写回填请求。在配对规则集中，添加算法组件并将回填优先级设置为 “高”。有关更多信息，请参阅 自定义匹配算法 (p. 13)。
3. 使用新的对战构建器数据更新游戏会话。使用服务器软件开发工具包回调函数使用对战信息更新您的游戏服务器。`onUpdateGameSession`（请参阅初始化服务器进程)。将代码添加到游戏服务器，在回填活动后处理更新的游戏会话对象。在 更新游戏服务器上的匹配数据 (p. 46) 中了解更多信息。
4. 关闭游戏会话的自动回填。您可以选择在单个游戏会话的任一时刻停止自动回填。要停止自动回填，请向您的游戏客户端或游戏服务器添加代码以进行 GameLift API 调用。StopMatchmaking. 此调用需要票证 ID。使用最新回填请求中的回填票证 ID。您可以从游戏会话对战数据中获取此信息，这些数据会按上一步中所述进行更新。

发送回填请求（来自游戏服务器）

您可以直接从托管游戏会话的游戏服务器进程发出对战回填请求。该服务器进程具有有关已连接到游戏的当前玩家及空余玩家位置状态的最新信息。

本主题假定您已构建必需的 FlexMatch 组件并已将对战过程成功地添加到您的游戏服务器和客户端游戏服务。有关设置 FlexMatch 的更多详细信息，请参阅FlexMatch 与 GameLift 托管集成 (p. 7)。

要为您的游戏启用对战回填，请添加以下功能：

- 将对战回填请求发送到对战构建器并跟踪请求的状态。
- 更新游戏会话的对战信息。请参阅 更新游戏服务器上的匹配数据 (p. 46)。

与其他服务器功能一样，游戏服务器使用 Amazon GameLift 服务器软件开发工具包。此软件开发工具包在 C++ 和 C# 中可用。

要从您的游戏服务器提出对战回填请求，请完成以下任务。

1. 触发对战回填请求。通常，每当已对战的游戏具有一个或多个空余玩家位置时，您就会想要发出回填请求。您可能希望将回填请求绑定到特定情况，例如填充关键人物角色或平衡团队。您还有可能想要基于游戏会话的时长限制回填活动。
2. 创建回填请求。添加代码以创建对战回填请求并将其发送到 FlexMatch 对战构建器。回填请求是使用以下服务器 API 处理的：

   - StartMatchBackfill()
   - StopMatchBackfill()

   要创建回填请求，请使用以下信息调用 `StartMatchBackfill`。要取消回填请求，请使用回填请求的票证 ID 调用 `StopMatchBackfill`。

   - 票证 ID— 提供对战票证 ID (或者选择自动生成该 ID)。您可以使用相同的机制来将票证 ID 分配到对战请求和回填请求。以相同方式处理对战和回填的票证。
   - 对战构建器— 确定要用于回填请求的对战。通常，您想要使用与用于创建原始对战的对战构建器相同的构建器。此请求需要对战配置 ARN。此信息存储在游戏会话对象中（GameSession)，该软件在激活游戏会话时由 Amazon GameLift 提供给服务器进程。对战配置 ARN 包含在 `MatchmakerData` 属性中。
   - 游戏会话 ARN— 确定要回填的游戏会话。您可以通过调用服务器 API 来获取游戏会话 ARN。GetGameSessionId(). 对战过程期间，新请求的票证不具有游戏会话 ID，而回填请求的票证则具有。提供游戏会话 ID 是用于区分新对战票证和回填票证的一种方式。

- 玩家数据— 包括玩家信息 (玩家) 对于您正在回填的游戏会话中的所有当前玩家。此信息让对战构建器能够为当前游戏会话中的玩家找到可能的最佳玩家匹配。您必须包括每位玩家的团队成员资格。如果你没有使用回填，请不要指定团队。如果您的游戏服务器已准确报告玩家连接状态，则您应能够获取此数据，如下所示：
  1. 托管游戏会话的服务器进程应具有玩家当前已连接到游戏会话的最新信息。
  2. 要获取玩家 ID、属性和团队任务，请从游戏会话对象中提取玩家数据 (GameSession),`MatchmakerData`属性（请参阅使用对战构建器数据 (p. 42))。对战构建器数据包含已与游戏会话匹配的所有玩家，因此您将需要提取仅当前连接的玩家的玩家数据。
  3. 对于玩家延迟，如果对战构建器调用延迟数据，则从所有当前玩家中收集新的延迟值并将其包含在每个 `Player` 对象中。如果忽略延迟数据而且对战构建器具有延迟规则，则该请求将不会匹配成功。回填请求仅需要游戏会话当前所在的区域的延迟数据。您可以从 `GameSession` 对象的 `GameSessionId` 属性中获取游戏会话的区域；此值是一个 ARN，其中包含了区域。

3. 跟踪回填请求的状态。使 Amazon GameLift 服务器软件开发工具包回调函数针对回填请求的状态更新您的游戏服务器。`onUpdateGameSession`（请参阅初始化服务器进程)。添加代码以在中处理状态消息以及由于回填请求成功更新的游戏会话对象。更新游戏服务器上的匹配数据 (p. 46).

   对战构建器在一个游戏会话中一次只能处理一个对战回填请求。如果您需要取消请求，请调用。StopMatchBackfill(). 如果您需要更改请求，请调用 `StopMatchBackfill`，然后提交更新的请求。

发送回填请求（来自客户服务）

作为从游戏服务器发送回填请求的替代方案，您可能希望从客户端游戏服务发送这些请求。要使用此选项，客户端服务必须有权访问有关游戏会话活动和玩家连接的最新数据；如果您的游戏使用会话目录服务，这可能是很好的选择。

本主题假定您已构建必需的 FlexMatch 组件并已将对战过程成功地添加到您的游戏服务器和客户端游戏服务。有关设置 FlexMatch 的更多详细信息，请参阅FlexMatch 与 GameLift 托管集成 (p. 7)。

要为您的游戏启用对战回填，请添加以下功能：

- 将对战回填请求发送到对战构建器并跟踪请求的状态。
- 更新游戏会话的对战信息。请参阅 更新游戏服务器上的匹配数据 (p. 46)

与其他客户端功能一样，客户端游戏服务使用Amazon使用 Amazon GameLift API。C++、C# 和许多其他语言都提供此软件开发工具包。有关客户端 API 的常规说明，请参阅 Amazon GameLift 服务 API 参考，其中介绍了 Amazon GameLift 相关操作的低级别服务 API，并提供特定于语言的参考指南链接。

要设置客户端游戏服务以回填对战的游戏，请完成以下任务。

1. 触发回填请求。通常，每当已对战的游戏具有一个或多个空余玩家位置时游戏都会启动回填请求。您可能希望将回填请求绑定到特定情况，例如填充关键人物角色或平衡团队。您还有可能想要基于游戏会话的时长限制回填。无论您对触发器使用什么，至少您将需要以下信息。您可以通过使用游戏会话 ID 调用 DescribeGameSessions 来从游戏会话对象 (GameSession) 中获取此信息。

   - 当前空余玩家位置的数量。此值可通过游戏会话的最大玩家限制和当前玩家数量计算得出。当前玩家数量会在您的游戏服务器每次连接 Amazon GameLift 服务时进行更新以验证新的玩家连接或报告断开连接的玩家。

   - 创建策略。此设置指示游戏会话当前是否接受新玩家。

   游戏会话对象包含其他可能有用的信息，包括游戏会话开始时间、自定义游戏属性和对战构建器数据。
2. 创建回填请求。添加代码以创建对战回填请求并将其发送到 FlexMatch 对战构建器。使用这些客户端 API 处理回填请求：

- StartMatchBackfill
- StopMatchmaking

要创建回填请求，请使用以下信息调用 `StartMatchBackfill`。回填请求类似于对战请求 (请参阅请求玩家的对战 (p. 39))，但还识别现有游戏会话。要取消回填请求，请使用回填请求的票证 ID 调用 `StopMatchmaking`。

- 票证 ID— 提供对战票证 ID (或者选择自动生成该 ID)。您可以使用相同的机制来将票证 ID 分配到对战请求和回填请求。以相同方式处理对战和回填的票证。
- 对战构建器— 确定要使用的对战配置的名称。通常，您想要使用与用于创建原始对战的回填的对战构建器相同的构建器。此信息位于对战配置 ARN 下的游戏会话对象 (GameSession), `MatchmakerData` 属性中。名称值是紧接在“matchmakingconfiguration/”之后的字符串。(例如，在 ARN 值“arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MM-4v4”中，对战配置名称为“MM-4v4”。)
- 游戏会话 ARN— 指定要回填的游戏会话。使用游戏会话对象中的 `GameSessionId` 属性；此 ID 使用您所需的 ARN 值。回填请求的对战票证 (MatchmakingTicket) 在进行处理时具有游戏会话 ID；在放置对战之前，新对战请求的票证不会获取游戏会话 ID；提供游戏会话 ID 是用于区分新对战票证和回填票证的一种方式。
- 玩家数据— 包括玩家信息 (玩家) 对于您正在回填的游戏会话中的所有当前玩家。此信息让对战构建器能够为当前游戏会话中的玩家找到可能的最佳玩家匹配。您必须包括每位玩家的团队成员资格。如果你没有使用回填，请不要指定团队。如果您的游戏服务器已准确报告玩家连接状态，则您应能够获取此数据，如下所示：

  1. 使用游戏会话 ID 调用 DescribePlayerSessions() 来发现当前已连接到游戏会话的所有玩家。每个玩家会话包括一个玩家 ID。您可以添加状态筛选器以仅检索活动的玩家会话。
  2. 从游戏会话对象 (GameSession)、`MatchmakerData` 属性中提取玩家数据 (请参阅使用对战构建器数据 (p. 42))。使用在上一步中获取的玩家 ID 来仅获取当前已连接玩家的数据。由于玩家退出时不会更新对战构建器数据，因此您仅需要提取当前玩家的数据。
  3. 对于玩家延迟，如果对战构建器调用延迟数据，请从所有当前玩家中收集新的延迟值并将其包含在 `Player` 对象中。如果忽略延迟数据而且对战构建器具有延迟规则，则该请求将不会匹配成功。回填请求仅需要游戏会话当前所在的区域的延迟数据。您可以从 `GameSession` 对象的 `GameSessionId` 属性中获取游戏会话的区域；此值是一个 ARN，其中包含了区域。

3. 跟踪回填请求的状态。添加代码以侦听对战票证状态更新。您可以使用设置的机制利用事件通知 (首选) 或轮询跟踪新对战请求的票证 (请参阅跟踪对战事件 (p. 40))。尽管您无需使用回填请求触发玩家接受活动，而且玩家信息已在游戏服务器上更新，但仍需要监控票证状态以处理请求失败和重新提交。

   对战构建器在一个游戏会话中一次只能处理一个对战回填请求。如果您需要取消请求，请调用 StopMatchmaking。如果您需要更改请求，请调用 `StopMatchmaking`，然后提交更新的请求。

   在对战回填请求成功后，您的游戏服务器会收到更新的 `GameSession` 对象并处理将新玩家加入游戏会话中所需的任务。请在更新游戏服务器上的匹配数据 (p. 46)上查看更多信息。

更新游戏服务器上的匹配数据

无论在您的游戏中如何启动对战回填请求，您的游戏服务器都必须能够处理由于对战回填请求而导致 Amazon GameLift 提供的游戏会话更新。

当 Amazon GameLift 完成对战回填请求时（无论成功与否），它都会使用回调函数调用您的游戏服务器。`onUpdateGameSession`. 此调用具有三个输入参数：对战回填票证 ID、状态消息和包含最新对战数据 (包括玩家信息) 的 GameSession 对象。您需要将以下代码添加到游戏服务器以作为您的游戏服务器集成的一部分：

1. 实现 `onUpdateGameSession` 函数。此函数必须能够处理以下状态消息 (`updateReason`)：

- MATCHMAKING_DATA_UPDATED — 新玩家已与游戏会话成功匹配。`GameSession` 对象包含更新的对战构建器数据，包括有关现有玩家和新匹配的玩家的玩家数据。
- BACKFILL_FAILED — 对战回填尝试由于内部错误而失败。`GameSession` 对象保持不变。
- BACKFILL_TIMED_OUT — 对战构建器未能在时间限制内找到回填对战。`GameSession` 对象保持不变。
- BACKFILL_CANCELLED — 对战回填请求已通过调用 StopMatchmake (客户端) 或 StopMatchBackfill (服务器) 而被取消。`GameSession` 对象保持不变。

2. 对于成功的回填对战，请使用更新的对战构建器数据来在新玩家连接到游戏会话时进行处理。至少，您将需要使用新玩家的团队任务以及要让玩家在游戏中开始所需的其他玩家属性。
3. 在您的游戏服务器调用服务器软件开发工具包操作ProcessReady()添加`onUpdateGameSession`回调方法名称作为进程参数。

FlexMatch 参考

本部分包含 GameLift FlexMatch 进行对战的参考文档。

主题

GameLift FlexMatch API 参考 (AmazonSDK)

本主题为的API操作提供了基于任务的列示 GameLift FlexMatch. 这些区域有： GameLift FlexMatch服务 API 已打包到Amazon中的开发工具包`aws.gamelift`命名空间。下载Amazon开发工具包要么查看亚马逊 GameLift API 参考文档.

GameLift FlexMatch 提供配对服务，用于托管的游戏 GameLift托管解决方案（包括自定义游戏服务器或实时服务器的托管托管，以及在 Amazon EC2 上托管 GameLift FleetIQ）以及其他托管系统，例如 peer-to-peer、本地或云计算基元。请参阅GameLift 开发人员指南了解有关其他的更多信息 GameLift 托管选项。

设置对战规则和流程

调用这些操作来创建 FlexMatch Matchmaker，为游戏配置对战流程，并为创建对战和团队定义一组用于创建对战和团队的自定义规则。

对战配置

- CreateMatchmakingConfiguration— 创建对战配置，其中包含评估玩家组和组建玩家团队的说明。使用 GameLift 对于托管，还要指定如何为对战创建新的游戏会话的方法。
- DescribeMatchmakingConfigurations— 检索定义的对战配置 GameLift 区域。
- UpdateMatchmakingConfiguration—更改对战配置的设置。队列。
- DeleteMatchmakingConfiguration— 从区域中删除对战配置。

对战规则集

- CreateMatchmakingRuleSet— 创建一组在搜索玩家匹配时使用的规则。
- DescribeMatchmakingRuleSets— 检索在中定义的对战规则集 GameLift 区域。
- ValidateMatchmakingRuleSet— 验证一组对战规则的语法。
- DeleteMatchmakingRuleSet—从区域中删除对战规则集。

为一名或多名玩家申请比赛

从游戏客户端服务调用这些操作来管理玩家对战请求。

- StartMatchmaking— 为一个玩家或想要参加同一对战的一组玩家请求对战。
- DescribeMatchmaking— 获取有关对战请求的详细信息，包括状态。

- AcceptMatch— 对于需要玩家接受的比赛，请通知 GameLift 当玩家接受建议的比赛时。
- StopMatchmaking— 取消对战请求。
- StartMatchBackfill-请求其他玩家匹配以填充现有游戏会话中的空位。

可用编程语言

这些区域有：Amazon支持的开发工具包 GameLift 具有以下语言。有关针对开发环境的支持的信息，请参阅每种语言的文档。

- C++ (软件开发工具包文档) (GameLift)
- Java (软件开发工具包文档) (GameLift)
- .NET (软件开发工具包文档) (GameLift)
- Go (软件开发工具包文档) (GameLift)
- Python (软件开发工具包文档) (GameLift)
- Ruby (软件开发工具包文档) (GameLift)
- PHP (软件开发工具包文档) (GameLift)
- JavaScript/Node.js (软件开发工具包文档) (GameLift)

FlexMatch 规则语言

本节中的参考主题描述了用于构建与 GameLift FlexMatch 配对规则的语法和语义。有关编写配对规则和规则集的详细帮助，请参阅构建 FlexMatch 规则集 (p. 12).

主题

FlexMatch 规则集架构

对 FlexMatch 规则集对小对战规则和大对战规则使用标准模式。有关每个部分的详细描述，请参阅FlexMatch 规则集属性定义 (p. 51).

小型比赛的规则集模式

以下架构记录了用于构建多达 40 名玩家的比赛的规则集的所有可能属性和允许的值。

```
{
    "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">
        }
    }],
    "expansions": [{
        "target": "string",
        "steps": [{
```

```
            "waitTimeSeconds": number,
            "value": number
        }, {
            "waitTimeSeconds": number,
            "value": number
        }]
    }]
}
```

大型匹配的规则集架构

以下架构记录了用于构建有 40 位以上玩家的对战的规则集的所有可能属性和允许的值。如果总`maxPlayers`规则集中所有团队的值超过 40，然后根据大型对战指南处理使用此规则集的对战请求。

```
{
    "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 规则集属性定义

本节定义了规则集架构中的每个属性。有关创建规则集的其他帮助，请参阅构建 FlexMatch 规则集 (p. 12).

name

规则集的描述性标签。此值与分配给 GameLift 的名称无关MatchmakingRuleSet 资源. 此值包含在描述已完成比赛的匹配数据中，但任何 GameLift 进程都没有使用它。

允许的值：字符串

必填？ 否

ruleLanguageVersion

正在使用的 FlexMatch 属性表达式语言的版本。

允许的值：“1.0”

必填？ 是

playerAttributes

配对请求中包含并在配对过程中使用的玩家数据的集合。您也可以在此声明属性，以便将玩家数据包含在传递给游戏服务器的匹配数据中，即使这些数据未在配对过程中使用。

必填？ 否

name

媒人使用的玩家属性的唯一名称。此名称必须与匹配请求中引用的玩家属性名称匹配。

允许的值：字符串

必填？ 是

type

玩家属性值的数据类型。

允许的值：“字符串”、“数字”、“string_list”、“string_number _map”

必填？ 是

default

对战请求未为玩家提供值时要使用的默认值。

允许的值：玩家属性允许的任何值。

必填？ 否

algorithm

可选配置设置，用于自定义配对过程。

必填？ 否

strategy

构建匹配项时使用的方法。如果未设置该属性，默认行为为为 “EefeSearch”。

允许的值：

- “RefeFeSearch” — 标准匹配方法。FlexMatch 通过基于一组自定义匹配规则评估池中的其他票证来围绕批次中最早的票证形成匹配。此策略用于 40 名或更少玩家的比赛。在使用此策略时，`batchingPreference`应该设置为 “随机” 或 “排序”。
- “平衡” — 经过优化以快速形成大型比赛的方法。该策略仅用于 41 至 200 名玩家的比赛。它通过对票池进行预先排序、建立潜在的比赛并将球员分配给球队，然后使用指定的玩家属性平衡比赛

中的每个球队来形成比赛。例如，该策略可用于平衡比赛中所有球队的平均技能水平。在使用此策略时，balancedAttribute必须设置，并batchingPreference应设置为 “最大人口” 或 “最快地区”。此策略无法识别大多数自定义规则类型。

必填？ 是

batchingPreference

在对门票进行分组以建立比赛之前使用的预先排序方法。对门票池进行预先排序会导致根据特定特征对门票进行批处理，这往往会提高最后一场比赛中玩家之间的统一性。

允许的值：

- “随机” — 仅适用于strategy= “详尽搜索”。没有进行预排序；游泳池中的门票是随机批量的。这是详尽搜索策略的默认行为。
- “已排序” — 仅适用于strategy= “详尽搜索”。票池是根据中列出的玩家属性预先排序的sortbyAttributes.
- “最大人口” — 仅适用于strategy= “平衡”。门票池按玩家报告可接受的延迟水平的区域进行预先排序。这是平衡策略的默认行为。
- “最快地区” — 仅适用于strategy= “平衡”。门票池按玩家报告其最低延迟水平的区域进行预先排序。结果比赛需要更长的时间才能完成，但所有玩家的延迟往往都很低。

必填？ 是

balancedAttribute

使用平衡策略构建大型比赛时要使用的玩家属性的名称。

允许的值：中声明的任何属性playerAttributes和type= “数字”。

必填？ 是的，如果strategy= “平衡”。

sortByAttributes

在批处理之前对票池进行预先排序时要使用的玩家属性列表。此属性仅在使用详尽的搜索策略进行预排序时使用。属性列表的顺序决定了排序顺序。FlexMatch 对字母和数字值使用标准排序约定。

允许的值：中声明的任何属性playerAttributes.

必填？ 是的，如果batchingPreference= “已排序”。

backfillPriority

匹配回填票的优先级方法。此属性决定 FlexMatch 何时批处理回填票证。它仅在使用详尽的搜索策略进行预分类时使用。如果未设置该属性，默认行为为为 “正常”。

允许的值：

- “普通” — 在组建匹配时不考虑票证的请求类型（回填或新匹配）。
- “高” — 票证批次按请求类型（然后按年龄）排序，FlexMatch 首先尝试匹配回填票证。
- “低” — TT 批次按请求类型（然后按年龄）排序，FlexMatch 首先尝试匹配非回填票证。

必填？ 否

expansionAgeSelection

计算匹配规则扩展等待时间的方法。如果比赛在一段时间过去之后尚未完成，则使用扩展来放宽比赛要求。等待时间是根据已经在部分填写的比赛中的门票的年龄计算的。如果未设置该属性，默认行为为为 “最新”。

允许的值：

- “最新” — 扩展等待时间是根据部分完成的比赛中具有最近创建时间戳的票证计算的。扩展的触发速度往往较慢，因为一张较新的票证可以重新启动等待时间时钟。

- “最旧” — 扩展等待时间是根据匹配中创建时间戳最早的票证计算的。扩张往往更快地触发。

必填？ 否

teams

对战游戏团队的配置。为每个团队提供团队名称和规模范围。规则集必须至少定义一个团队。

name

团队的唯一名称。规则和扩展可以引用团队名称。在成功的比赛中，球员将在配对数据中按球队名称分配。

允许的值：字符串

必填？ 是

maxPlayers

可以分配给团队的玩家的最大数量。

允许的值：Number

必填？ 是

minPlayers

在对战可行之前必须分配给团队的玩家的最小数量。

允许的值：Number

必填？ 是

quantity

在比赛中要创建的此类球队的数量。数量大于 1 的团队将使用附加编号（“red_1”、“red_2” 等）来指定。如果未设置该属性，默认值为 “1”。

允许的值：Number

必填？ 否

rules

规则语句集，此类语句用于定义如何评估对战游戏的玩家。

必填？ 否

name

规则的唯一名称。规则集中的所有规则必须具有唯一的名称。规则名称在跟踪与规则相关的活动的事件日志和指标中引用。

允许的值：字符串

必填？ 是

description

规则的文本描述。此信息可用于标识规则的用途。对战过程未使用该选项。

允许的值：字符串

必填？ 否

type

规则语句的类型。每种规则类型都有必须设置的其他属性。有关每种规则类型的结构和用法的更多详细信息，请参阅FlexMatch 规则类型 (p. 55).

允许的值：

- “AbsoluteSort” — 使用明确的排序方法进行排序，该方法根据指定的玩家属性是否与批次中最旧的票证进行比较，对门票进行批次排序。
- “集合” — 评估集合中的值，例如作为集合的玩家属性或多个玩家的一组值。
- “比较” — 比较两个值。
- “距离” — 测量数字值之间的距离。
- “BatchDistance” — 衡量属性值之间的差异并使用它对匹配请求进行分组。
- “DistanceSort” — 使用显式排序方法进行排序，该方法根据具有数值的指定玩家属性与批次中最早的票证的比较，对一批票进行排序。
- “延迟” — 评估为匹配请求报告的区域延迟数据。

必填？ 是

expansions

在比赛无法完成时随着时间的推移放宽比赛要求的规则。将扩展设置为一系列逐步应用的步骤，以便更容易找到比赛。默认情况下，FlexMatch 会根据添加到匹配中的最新票证的时间计算等待时间。您可以使用算法属性更改扩展等待时间的计算方式`expansionAgeSelection`.

扩展等待时间是绝对值，因此每个步骤的等待时间都应比上一步长。例如，要计划一系列渐进的扩展，可以使用 30 秒、40 秒和 50 秒的等待时间。等待时间不能超过对战请求允许的最长时间，此时间在对战配置中设置。

必填？ 否

target

要放宽的规则集元素。你可以放松团队规模属性或任何规则声明属性。语法是 "<component name>[<rule/team name>]。 <property name>”。例如，要更改团队的最小规模：`teams[Red, Yellow].minPlayers`. 要在名为 “minSill” 的比较规则语句中更改最低技能要求：`rules[minSkill].referenceValue`.

必填？ 是

steps

waitTimeSeconds

为目标规则集元素应用新值之前等待的时间长度，以秒为单位。

必填？ 是

value

目标规则集元素的新值。

FlexMatch 规则类型

Batch 规则规则

```
batchDistance
```

Batch 距离规则用于测量两个属性值之间的差异。您可以将批量距离规则类型用于大匹配和小匹配。共有两种类型的批距离规则：

- 比较数值属性值. 例如，某项类型的批处理距离规则可能要求对战游戏的所有玩家必须在相邻的两个技能级别内。对于此类型，定义一个最大距离batchAttribute所有门票的。
- 比较字符串属性值. 例如，某项类型的批处理距离规则可能要求对战游戏的所有玩家必须请求相同的游戏模式。对于此类型，定义一个batchAttributeValue T FlexMatch 用于形成批处理。

Batch 规则属性

- **batchAttribute**— 用于形成批处理的玩家属性值。
- **maxDistance**— 为了成功进行对战游戏而允许的最大距离值。用于比较数值属性。
- **partyAggregation**— 决定方式的值 FlexMatch 处理多个玩家（派对）的门票。有效选项包括最小值的选项 (min)，最大值 (max) 和平均值 (avg) 门票玩家的值。默认为 avg。

Example

示例

```
{
  "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"
}
```

比较规则

```
comparison
```

比较规则将一个玩家属性值与另一个值进行比较。有两种类型的比较规则：

- 与参考值比较. 例如，一个对战构建器的比较规则可能要求匹配的玩家具有特定技能水平。对于此类型，请指定玩家属性、参考值和比较操作。
- 在玩家之间进行比较. 例如，某项对战构建器的比较规则可能要求对战游戏的所有玩家必须使用不同的角色。对于此类型，请指定玩家属性和 equal (=) 或不等于 (!=) 比较操作。不要指定参考值。

> **Note**
>
> Batch 距离规则在比较玩家属性时更有效。减少对接会潜伏, 尽可能使用批量距离规则.

比较规则属性

- **measurements**— 要比较的玩家属性值。
- **referenceValue**— 为了潜在对战游戏而将测量值与之进行比较。
- **operation**— 确定如何将测量值与参考值进行比较的值。有效操作包括：<、<=、=、!=、>、>=.
- **partyAggregation**— 决定方式的值 FlexMatch 处理多个玩家（派对）的门票。有效选项包括最小值的选项 (min)，最大值 (max) 和平均值 (avg) 门票玩家的值。默认为 avg。

距离规则

```
distance
```

距离规则计量两个数字值之间的差值，例如玩家技能级别之间的差距。例如，某项距离规则可能要求所有玩家必须玩游戏至少 30 个小时。

> **Note**
>
> Batch 距离规则在比较玩家属性时更有效。减少对接会潜伏, 尽可能使用批量距离规则.

距离规则属性

- **measurements**— 要为其度量距离的玩家属性值。这必须是具有数值的属性。
- **referenceValue**— 用于对照潜在对战游戏度量距离的数字值。
- **minDistance/maxDistance**— 为了成功进行对战游戏而的最小或最大距离的值。
- **partyAggregation**— 决定方式的值 FlexMatch 处理多个玩家（派对）的门票。有效选项包括最小值的选项 (min)，最大值 (max) 和平均值 (avg) 门票玩家的值。默认为 avg。

收集规则规则

```
collection
```

收集规则将一组玩家属性值与批次中其他玩家的属性值或参考值进行比较。一个集合可以包含多个玩家的属性值和/或采用字符串列表形式的一个玩家属性。例如，收集规则可能会查看队伍中玩家选择的角色。然后，该规则可能会要求队伍至少有一个特定角色。

收集规则属性

- **measurements**— 要比较的玩家属性值的集合。属性值必须采用字符串列表形式。
- **referenceValue**— 用于比较潜在对战游戏的测量值的Value（或值集合）。
- **operation**— 确定如何比较测量值集合的值。有效操作如下所示：
  - intersection— 此操作将计量所有玩家集合中的相同值的数量。有关使用交叉点运算的规则的示例，请参阅示例 4：使用显式排序查找最佳匹配 (p. 26).
  - contains— 此操作计量包含指定参考值的玩家属性集合的数量。有关使用 contains 操作的规则的示例，请参阅示例 3：设置团队级要求和延迟限制 (p. 24).
  - reference_intersection_count— 此操作测量玩家属性集合中与参考值集合中的物品匹配的物品数量。您可以使用此操作来比较多个不同的玩家属性。有关比较多个玩家属性集合的规则示例，请参阅示例 5：查找跨多个玩家属性的交集 (p. 28).
- **minCount/maxCount**— 为了成功进行对战游戏而的最小或最大计数值。
- **partyAggregation**— 决定方式的值 FlexMatch 处理多个玩家（派对）的门票。对于此值，您可以使用。union将所有玩家的玩家属性进行组合。或者，您可以使用。intersection使用队伍共有的玩家属性。默认为 union。

延迟规则

```
latency
```

延迟规则用于衡量每个位置的玩家延迟。延迟规则将忽略延迟高于最大延迟规则的任何位置。玩家的延迟值必须低于至少一个位置的最大值，延迟规则才能接受它们。您可以将此规则类型用于大型匹配项，方法是指定maxLatency财产。

延迟规则属性

- **maxLatency**— 位置的最大可接受延迟值。如果票证没有延迟低于最大值的位置，则该票证与延迟规则不匹配。
- **maxDistance**— 每张票的延迟与距离参考值之间的最大值。
- **distanceReference**— 用于比较票证延迟的延迟值。距离参考值最大距离内的门票将导致匹配成功。有效选项包括最小值的选项 (`min`) 和平均值 (`avg`) 玩家延迟值。
- **partyAggregation**— 决定方式的值 FlexMatch 处理多个玩家（派对）的门票。有效选项包括最小值的选项 (`min`)，最大值 (`max`) 和平均值 (`avg`) 门票玩家的值。默认为 `avg`。

> **Note**
>
> 队列可以在与延迟规则不匹配的区域中放置游戏会话。有关队列的延迟策略的更多信息，请参阅。创建玩家延迟策略.

绝对排序规则规则

```
absoluteSort
```

绝对排序规则根据指定的玩家属性对一批对接会门票与添加到批次中的第一张票证进行比较.

绝对排序规则属性

- **sortDirection**—对配对门票进行排序的顺序. 有效选项包括`ascending`和`descending`.
- **sortAttribute**— 排序门票所依据的玩家属性。
- **mapKey**-用于对玩家的地图属性进行排序的选项。有效选项包括：
  - `minValue`— 值最低的密钥是第一个。
  - `maxValue`— 具有最高值的密钥是第一个。
- **partyAggregation**— 决定方式的值 FlexMatch 处理多个玩家（派对）的门票。有效选项包括最小值的选项 (`min`) 玩家属性，最大值 (`max`) 玩家属性，以及平均 (`avg`) 队伍中玩家的所有玩家属性。默认为 `avg`。

Example

示例

以下示例规则按技能等级对玩家进行排序，并平均各方的技能水平。

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

距离排序规则

```
distanceSort
```

距离排序规则根据指定玩家属性与添加到批次的第一张票证的距离对一批配对门票进行排序.

距离排序规则属性

- **sortDirection**—排序方向对接会车票. 有效选项包括ascending和descending.
- **sortAttribute**— 排序门票所依据的玩家属性。
- **mapKey**-用于对玩家的地图属性进行排序的选项。有效选项包括：
  - minValue— 对于添加到批次中的第一个票证，找到包含最低值的关键字。
  - maxValue— 对于添加到批次中的第一个票证，找到包含最高值的关键字。
- **partyAggregation**— 决定方式的值 FlexMatch 处理多个玩家（派对）的门票。有效选项包括最小值的选项 (min)，最大值 (max) 和平均值 (avg) 门票玩家的值。默认为 avg。

FlexMatch 属性表达式

属性表达式可用于定义与对战有关的某些属性。它们允许您在定义属性值时使用计算和逻辑。属性表达式通常导致以下两种形式之一：

- 单个玩家数据。
- 计算出个人玩家数据的集合。

常见的匹配属性表达式

属性表达式标识玩家、团队或对战游戏的特定值。以下部分表达式说明了如何标识团队和玩家：

目标	输入	意义	输出
标识对战游戏的特定团队：	teams[red]	红队	团队
标识对战游戏的特定团队：	teams[red,blue]	红队和蓝队	List<Team>
标识对战游戏的所有团队：	teams[*]	所有团队	List<Team>
标识特定团队中的玩家：	team[red].players	红队中的玩家	List<Player>
标识对战游戏的特定团队中的玩家：	team[red,blue].players	对战游戏的玩家 (按团队分组)	List<List<Player>>
标识对战游戏的玩家：	team[*].players	对战游戏的玩家 (按团队分组)	List<List<Player>>

属性表达式示例

下表给出基于之前示例构建的部分属性表达式：

表达式	意义	结果类型
teams[red].players[playerid]	红队所有玩家的玩家 ID	List<string>
teams[red].players.attribute[skill]	红队所有玩家的“技能”属性	List<number>
teams[red,blue].players.attribute[skill]	对战游戏的所有玩家的 “技能” 属性 (按团队分组)	List<List<number>>

表达式	意义	结果类型
`teams[*].players.attributes[skill]`	对战游戏的所有玩家的“技能”属性 (按团队分组)	List<List<number>>

属性表达式聚合

属性表达式可用于使用以下函数或组合函数来聚合团队数据：

聚合	输入	意义	输出
`min`	List<number>	获取列表中所有数字的最小值。	number
`max`	List<number>	获取列表中所有数字的最大值。	number
`avg`	List<number>	获取列表中所有数字的平均值。	number
`median`	List<number>	获取列表中所有数字的中值。	number
`sum`	List<number>	获取列表中所有数字的总和。	number
`count`	List<?>	获取列表中的元素数量。	number
`stddev`	List<number>	获取列表中所有数字的标准差。	number
`flatten`	List<List<?>>	将嵌套列表的集合变成包含所有元素的单个列表。	List<?>
`set_intersection`	List<List<string>>	获取在集合的所有字符串列表中找到的字符串列表。	List<string>
以上全部	List<List<?>>	对嵌套列表的所有操作会对每个子列表执行一次以生成结果列表。	List<?>

下表给出使用聚合函数的部分有效属性表达式：

表达式	意义	结果类型
flatten(teams[*].players.attributes[skill])	对战游戏中的所有玩家的“技能”属性 (未分组)	List<number>
avg(teams[red].players.attributes[skill])	红队玩家的平均技能	number
avg (team [*] .players.attributes [技能])	对战游戏中的每个团队的平均技能	List<number>

表达式	意义	结果类型
avg(flatten(teams[*].players.attributes[skill]))	对战游戏中的所有玩家的平均技能级别。该表达式获取玩家技能的展开列表，然后计算它们的平均值。	number
count(teams[red].players)	红队的玩家数量	number
count (teams[*].players)	对战游戏中的每个团队的玩家数量	List<number>
max(avg(teams[*].players.attributes[skill]))	对战游戏中的最高团队技能级别	number

FlexMatch 对战事件

GameLift FlexMatch 会在处理过程中为每张配对票发出事件。您可以将这些事件发布到 Amazon SNS 主题，如中所述设置 FlexMatch 事件通知 (p. 35). 这些事件也会在尽最大努力的基础上几乎实时地发送到 Amazon CloudWatch Event。

本主题介绍 FlexMatch 事件的结构，并为每种事件类型提供了一个示例。有关对战票证状态的更多信息，请参阅MatchmakingTicket中的Amazon GameLift API 参考.

MatchmakingSearching

票证已输入到对战中。这包括新的请求和失败的建议对战游戏所包含的请求。

资源：ConfigurationArn

详细信息：类型、票证、estimatedWaitMillis、gameSessionInfo

示例

```
{
  "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

潜在的对战游戏已创建。这是对所有新潜在对战游戏发出的，不论是否需要接受。

资源：ConfigurationArn

详细信息：类型、票
证、acceptanceTimeout、acceptanceRequired、ruleEvaluationMetrics、gameSessionInfo、matchId

示例

```
{
  "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

玩家已接受潜在的对战游戏。此事件包含对战游戏中每个玩家的当前接受状态。缺少此数据意味着尚未给该玩家调用 AcceptMatch。

资源：ConfigurationArn

详细信息：类型、票证、matchId、gameSessionInfo

示例

```
{
  "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

一旦玩家接受、玩家拒绝或接受超时，对战游戏接受操作即完成。

资源：ConfigurationArn

详细信息：类型、票证、接受、matchId、gameSessionInfo

示例

```
{
  "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

对战匹配已成功完成并已创建游戏会话。

资源：ConfigurationArn

详细信息：类型、票证、matchId、gameSessionInfo

示例

```
{
  "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

对战票证由于超时而失败。

资源：ConfigurationArn

详细信息：类型、票证、ruleEvaluationMetrics、消息、matchId、gameSessionInfo

示例


```
{
  "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

对战票证已取消。

资源：ConfigurationArn

详细信息：类型、票证、ruleEvaluationMetrics、消息、matchId、gameSessionInfo

示例

```
{
```

```
  "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

对战票证遇到了错误。这可能是由于无法访问游戏会话队列或内部错误所致。

资源：ConfigurationArn

详细信息：类型、票证、ruleEvaluationMetrics、消息、matchId、gameSessionInfo

示例


```
{
  "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"
  }
}
```

使用 FlexMatch 对战

Amazon 的云安全性的优先级最高。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 Amazon 客户，您也将从这些数据中心和网络架构受益。

安全性是 Amazon 和您的共同责任。有关如何在使用 FlexMatch 时应用责任共担模式，请参阅Amazon GameLift 中的安全性.

GameLift FlexMatch 发行说明和 SDK 版本

GameLift 发布说明提供与服务相关的新功能、更新和修复的详细信息。此页面还包括 GameLift SDK 版本历史记录。

GameLift 开发者资源

要查看所有 GameLift 文档和开发人员资源，请参阅Amazon GameLift 文档主页。

Amazon词汇表

有关最新Amazon术语，请参阅《Amazon一般参考》中的Amazon术语表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。