FlexMatch 开发人员指南

Amazon GameLift

版本

Amazon GameLift: FlexMatch 开发人员指南

Table of Contents

什么是 Amazon GameLift FlexMatch？

Amazon GameLift FlexMatch 是一项针对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，定义多人游戏对战的内容，并确定如何评估和选择每场对战的匹配玩家。您还可以根据自己的游戏需求微调对战算法的关键方面。

将 FlexMatch 作为独立的对战服务使用，或将其与 Amazon GameLift 游戏托管解决方案集成使用。例如，您可以将 FlexMatch 作为一项独立功能，用于采用点对点架构的游戏或使用其他云计算解决方案的游戏。或者，您也可以将 FlexMatch 添加到 Amazon GameLift 托管式 EC2 托管或添加到使用 Amazon GameLift Anywhere 的本地托管。本指南详细介绍了如何针对特定场景构建 FlexMatch 对战系统。

FlexMatch 使您可以根据自己的游戏要求灵活地设置对战优先级。例如，您可以执行以下操作：

- 在对战速度和质量之间找到平衡。设置对战规则以快速找到足够好的对战，或者让玩家等待更长的时间才能找到最佳的对战，以获得最佳的玩家体验。
- 根据匹配良好的玩家或匹配良好的团队进行对战。创建所有玩家都具有相似特征（如技能或经验）的对战。或者，您也可以创建每个团队的综合特征符合共同标准的对战。
- 优先考虑玩家延迟因素如何影响对战。您是想对所有玩家设置延迟硬性限制，还是只要对战中每个人的延迟都差不多，就可以接受更高的延迟？

> **准备好开始使用 FlexMatch 了吗？**
>
> 有关在 FlexMatch 上启动和运行游戏的分步指南，请参阅以下主题：
>
> - 教程：将 FlexMatch 与 Amazon GameLift 托管集成
> - 教程：将 FlexMatch 与独立对战集成

FlexMatch 的主要功能

以下功能适用于所有 FlexMatch 场景，无论您是将 FlexMatch 用作独立服务还是使用 Amazon GameLift 游戏托管。

- 可自定义的玩家匹配。设计和建造对战构建器，以适应您为玩家提供的所有游戏模式。构建一组自定义规则以评估玩家属性 (如技能级别或角色) 并为游戏组建最可能的玩家对战。

- 基于延迟的匹配。提供玩家延迟数据并创建匹配规则，要求对战中的玩家具有相似的响应时间。当您的玩家对战池跨越多个地理区域时，此功能非常有用。
- Support 支持最多 200 名玩家的对战规模。使用为您的游戏定制的对战规则，创建最多 40 名玩家的对战。使用匹配流程创建多达 200 名玩家的对战，该流程使用简化的自定义匹配流程来控制玩家的等待时间。
- 玩家接受 要求玩家在完成对战并开始游戏会话之前选择加入拟议的对战。使用此功能可以启动您的自定义接受工作流程，并在为对战设置新的游戏会话之前向 FlexMatch 报告玩家的回应。如果不是所有玩家都接受对战，则提议的对战将失败，接受对战的玩家将自动返回对战池。
- 支持玩家组队。为要组队一起玩游戏的一组玩家生成对战游戏。查找更多玩家以根据需要填充对战。
- 可扩展的匹配规则。经过一定时间后仍未找到成功的匹配项，请逐渐放宽对战要求。规则扩展可以让您决定在何时何地放松最初的对战规则，这样玩家就可以更快地进入可玩的游戏。
- 匹配回填。在现有游戏会话中的可用玩家位置，填充现有游戏会话中的可用玩家位置。自定义何时以及如何请求新玩家，并使用相同的自定义匹配规则来寻找更多玩家。

使用 Amazon GameLift 托管 FlexMatch

FlexMatch 提供以下附加功能，供您用于要使用 Amazon GameLift 进行托管的游戏。这包括使用自定义游戏服务器或实时服务器的游戏。

- 游戏会话位置。成功进行对战后，FlexMatch 会自动向 Amazon GameLift 请求新的游戏会话放置。对战期间生成的数据，包括玩家 ID 和队伍分配，将提供给游戏服务器，以便它可以使用这些信息开始对战的游戏会话。然后，FlexMatch 会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对战中遇到的延迟，Amazon GameLift 的游戏会话放置也可以使用区域玩家延迟数据（如果提供）。
- 自动匹配回填。启用此功能后，当新的游戏会话开始时玩家空位时，FlexMatch 会自动发送匹配回填请求。您的对战系统以最少数量的玩家开始游戏会话放置过程，然后快速填补剩余的空位。您不能使用自动回填来替换退出匹配游戏会话的玩家。

如果您将 Amazon GameLift FleetIQ 用于使用 Amazon Elastic Compute Cloud（Amazon EC2）资源进行托管的游戏，请将 FlexMatch 作为独立服务实施。

Amazon GameLift FlexMatch 的定价

Amazon GameLift 按使用时长对实例收费，按传输的数据量收取带宽费用。如果您在 Amazon GameLift 服务器上托管游戏，则 Amazon GameLift 的费用中包含 FlexMatch 使用量。如果您在其他

服务器解决方案上托管游戏，FlexMatch 使用量将单独收费。有关 Amazon GameLift 的费用和价格的完整列表，请参阅 Amazon GameLift 定价。

有关计算托管游戏或通过 Amazon GameLift 对战的成本的信息，请参阅生成 Amazon GameLift 定价估算值，其中描述了如何使用 AWS Pricing Calculator。

Amazon GameLift FlexMatch 的工作原理

本主题概述了 Amazon GameLift FlexMatch 服务，包括 FlexMatch 系统的核心组件及其交互方式。

您可以将 FlexMatch 用于使用 Amazon GameLift 托管托管的游戏或使用其他托管解决方案的游戏。托管在 Amazon GameLift 服务器上的游戏（包括实时服务器）使用集成的 Amazon GameLift 服务来自动找到可用的游戏服务器并开始对战的游戏会话。使用 FlexMatch 作为独立服务的游戏，包括 Amazon GameLift FleetIQ，必须与现有托管系统协调以分配托管资源并开始对战的游戏会话。

有关为游戏设置 FlexMatch 的详细指南，请参阅开始使用 FlexMatch。

对战组件

FlexMatch 对战系统包括以下部分或全部组件。

Amazon GameLift 组件

这些是 Amazon GameLift 资源，用于控制 FlexMatch 服务如何为您的游戏进行对战。它们是使用 Amazon GameLift 工具（包括控制台和 AWS CLI）创建和维护的，或者也可以使用适用于 Amazon GameLift 的 AWS 软件开发工具包以编程方式创建和维护。

- FlexMatch 对战配置（也称为对战构建器）– 对战构建器是一组配置值，可为您的游戏自定义对战流程。一款游戏可以有多个对战构建器，根据需要为不同的游戏模式或体验进行配置。当您的游戏向 FlexMatch 发送对战请求时，它会指定要使用哪个对战构建器。
- FlexMatch对战规则集 – 规则集包含评估玩家是否有潜在匹配以及批准或拒绝所需的所有信息。规则集定义了对战的团队结构，声明了用于评估的玩家属性，并提供了描述可接受对战标准的规则。规则可以适用于单个玩家、团队或整个对战。例如，规则可能要求对战中的每位玩家选择相同的游戏地图，或者可能要求所有团队的玩家平均技能相似。
- Amazon GameLift 游戏会话队列（仅适用于使用 Amazon GameLift 托管托管的 FlexMatch）– 游戏会话队列查找可用的托管资源并为对战启动新的游戏会话。队列的配置决定了 Amazon GameLift 在哪里寻找可用托管资源以及如何为匹配选择最佳可用主机。

自定义组件

以下组件包含完整的 FlexMatch 系统所需的功能，您必须根据游戏架构实现这些功能。

- 用于对战的玩家界面 – 此界面允许玩家加入对战。它至少会通过客户端对战服务组件发起对战请求，并根据对战过程的需要提供玩家特定的数据，例如技能等级和延迟数据。

  > **Note**
  >
  > 作为最佳实操，与 FlexMatch 服务的通信应由后端服务完成，而不是通过游戏客户端进行。

- 客户对战服务 – 该服务从玩家界面发送玩家加入请求，生成对战请求，然后将其发送到 FlexMatch 服务。对于正在处理的请求，它会监控对战事件，跟踪对战状态，并根据需要采取行动。根据您在游戏中管理游戏会话托管的方式，此服务可能会将游戏会话连接信息返回给玩家。此组件使用带有 Amazon GameLift API 的 AWS 软件开发工具包与 FlexMatch 服务进行通信。
- 对战放置服务（仅适用于作为独立服务的 FlexMatch）– 此组件可与您现有的游戏托管系统配合使用，以查找可用的托管资源并为对战启动新的游戏会话。该组件必须获取对战结果并提取开始新游戏会话所需的信息，包括玩家 ID、属性和对战中所有玩家的队伍分配。

FlexMatch 对战流程

本主题描述了基本的对战场景以及您的各种游戏组件与 FlexMatch 服务之间的交互。

请求玩家对战

使用您的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对战服务向 FlexMatch 发送对战请求。该请求标识了在满足请求时要使用的 FlexMatch 对战构建器。该请求还包括您的自定义对战构建器所需的玩家信息，例如技能等级、游戏偏好或地理延迟数据。您可以为一个玩家或多个玩家提出对战请求。

将请求添加到对战池

当 FlexMatch 收到对战请求时，它会生成一张对战票证并将其添加到对战构建器的票证池中。票证会留在池中，直到达到对战构建器的时间限制。您的客户对战服务会定期收到有关对战活动的通知，包括票证状态的变化。

构建对战

您的 FlexMatch 对战构建器会持续对其池中的所有票证运行以下流程：

1. 对战构建器按票证时长对票证池进行排序，然后从最旧的票证开始建立潜在的匹配项。

2. 对战构建器为潜在匹配项添加第二张票证，并根据您的自定义对战规则评估结果。如果潜在的对战通过评估，则票证的玩家将被分配到一支队伍中。
3. 对战构建器按顺序添加下一张票证并重复评估过程。当所有玩家位置都填满后，对战就准备好了。

大型对战（41 至 200 名玩家）使用上述流程的修改版本，因此它可以在合理的时间范围内进行对战。对战构建器不是单独评估每张票证，而是将预先排序的票证池划分为潜在的对战，然后根据您指定的玩家特征来平衡每场对战。例如，对战构建器可能会根据相似的低延迟位置对票证进行预先排序，然后使用赛后平衡来确保各支队伍在玩家技能方面均匀匹配。

报告对战结果

找到可接受的匹配项后，将更新所有匹配的票证，并为每张匹配的票证生成成功的对战事件。

- FlexMatch 作为一项独立服务：在成功的对战活动中，您的游戏会收到匹配结果。结果数据包括所有匹配的玩家及其队伍分配的列表。如果您的对战请求包含玩家延迟信息，则结果还会建议对战的最佳地理位置。
- 使用 Amazon GameLift 托管解决方案进行灵活匹配：匹配结果会自动传递到 Amazon GameLift 队列以进行游戏会话放置。对战构建器决定使用哪个队列置放游戏会话。

开始对战的游戏会话

成功组建提议的对战后，将开始新的游戏会话。在为对战设置游戏会话时，您的游戏服务器必须能够使用对战结果数据，包括玩家 ID 和队伍分配。

- FlexMatch 作为一项独立服务：您的自定义对战放置服务会从成功的对战活动中获取匹配结果数据，并连接到您现有的游戏会话放置系统，为对战找到可用的托管资源。找到托管资源后，匹配放置服务会与您现有的托管系统进行协调，以启动新的游戏会话并获取连接信息。
- 使用 Amazon GameLift 托管解决方案进行 FlexMatch：游戏会话队列为对战找到可用的最佳游戏服务器。根据队列的配置方式，它会尝试将游戏会话置于成本最低的资源以及玩家将体验低延迟的地方（如果提供了玩家延迟数据）。成功放置游戏会话后，Amazon GameLift 服务会提示游戏服务器开始新的游戏会话，传递对战结果和其他可选的游戏数据。

将玩家与对战联系起来

游戏会话开始后，玩家连接到会话，领取队伍任务，然后开始玩游戏。

- FlexMatch 作为一项独立服务：您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。
- 使用 Amazon GameLift 托管解决方案进行 FlexMatch：成功放置游戏会话后，FlexMatch 会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。

支持 FlexMatch 的 AWS 区域

如果您将 FlexMatch 与 Amazon GameLift 托管解决方案结合使用，则可以在托管游戏的任何位置托管匹配的游戏会话。请参阅 Amazon GameLift 托管的 AWS 区域和位置的完整列表。

对于所有 FlexMatch 用户，您可以在以下支持的 AWS 区域中托管 FlexMatch 资源，包括对战配置和规则集。请参阅为对战构建器选择一个区域。

AWS 区域 名称	区域代码
美国东部（弗吉尼亚州北部）	us-east-1
美国西部（俄勒冈州）	us-west-2
亚太地区（首尔）	ap-northest-2
亚太地区（悉尼）	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
欧洲地区（法兰克福）	eu-central-1
欧洲地区（爱尔兰）	eu-west-1
中国（北京和宁夏区域）	

开始使用 FlexMatch

使用本节中的资源可以帮助您入门并使用FlexMatch构建对战系统。

主题

教程：设置 FlexMatch

Amazon GameLift FlexMatch AWS 是一项服务，您必须AWS有一个账户才能使用这项服务。创建 AWS 账户是免费的。有关如何使用 AWS 账户的更多信息，请参阅 AWS 入门。

如果您将 FlexMatch 与其他 Amazon GameLift 解决方案一起使用，请参阅以下主题：

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

为 Amazon GameLift 设置账户

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 AWS Identity and Access Management (IAM) 服务控制台，按照步骤创建或更新用户或用户组。IAM 管理对您的AWS服务和资源的访问权限。所有使用 Amazon GameLift 控制台或调用 Amazon GameLift API 访问您的 FlexMatch 资源的人都必须获得明确的访问权限。有关使用控制台（或 AWS CLI 或其他工具）设置用户组的详细说明，请参阅创建 IAM 用户。
3. 将权限策略附加到账户中的用户或组？通过向用户或用户组附加 IAM 策略来管理对AWS服务和资源的访问权限。权限策略指定了用户必须有权访问的一组AWS服务和操作。

   对于 Amazon 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。使用以下示例来创建您的策略。

以下示例说明了一种内联权限策略，该策略具有所有 Amazon GameLift 资源和操作的管理权限。您可以通过仅指定 FlexMatch 特定的项目来选择限制访问权限。

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

教程：将 FlexMatch 与独立对战集成

本主题概述了将 FlexMatch 作为独立对战服务实现的完整集成过程。如果您的多人游戏使用点对点、自定义配置的本地硬件或其他云计算基元托管，请使用此流程。此过程也适用于 Amazon GameLift FleetIQ，这是一款适用于托管在 Amazon EC2 上的游戏的托管优化解决方案。如果您使用 Amazon GameLift 托管托管资源（包括实时服务器）托管游戏，请参阅教程：将 FlexMatch 与 Amazon GameLift 托管集成。

在开始集成之前，您必须拥有一个AWS账户并设置Amazon GameLift服务的访问权限。有关详细信息，请参阅教程：设置 FlexMatch。与创建和管理 Amazon GameLift FlexMatch 对战构建器和规则集有关的所有基本任务都可以使用 Amazon GameLift 控制台完成。

1. 创建对战规则集 您的自定义规则集提供了有关如何构造匹配项的完整说明。在其中，您可以定义每个团队的结构和规模。您还提供了一组对战必须满足才能生效的要求，FlexMatch 使用这些要求来包括或排除对战中的玩家。这些要求可能适用于个人玩家。您还可以在规则集中自定义 FlexMatch 算法，例如创建最多可容纳 200 名玩家的大型对战。请参阅以下主题：

   - 构建 FlexMatch 规则集
   - FlexMatch 规则集示例

2. 设置 事件通知。使用通知来跟踪 FlexMatch 对战活动，包括待处理的匹配请求的状态。这是用来提供拟议匹配结果的机制。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。通知是首选选项。请参阅以下主题：

   - 设置 FlexMatch 事件通知
   - FlexMatch 对战活动

3. 设置 FlexMatch 对战配置。也称为对战构建器，此组件接收对战请求并进行处理。您可以通过指定规则集、通知目标和最长等待时间来配置对战构建器。也可以启用可选功能。请参阅以下主题：

- 设计 FlexMatch 对战构建器
- 创建对战配置

4. 建立客户对战服务。创建或扩展游戏客户端服务，该服务具有构建和向 FlexMatch 发送对战请求的功能。要生成对战请求，此组件必须具有获取对战规则集所需的玩家数据的机制，以及区域延迟信息（可选）。它还必须有一种方法来为每个请求创建和分配唯一的票证 ID。您也可以选择建立玩家接受工作流程，要求玩家选择加入提议的对战。该服务还必须监控对战事件以获取对战结果，并启动游戏会话放置以成功对战。请参阅以下主题：

   - 将 FlexMatch 添加到游戏客户端

5. 建立匹配放置服务。创建一种可与现有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对战。该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开始新的游戏会话。您可能还需要实现一个工作流程来发出匹配回填请求，该工作流程使用对战来填补已经在运行的匹配游戏会话中的空缺位置。

教程：将 FlexMatch 与 Amazon GameLift 托管集成

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

- 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。请参阅以下主题：
    - 构建 FlexMatch 规则集
    - FlexMatch 规则集示例
  - 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
  - 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。请参阅以下主题：
    - 设计 FlexMatch 对战构建器

    - 创建对战配置
- 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：
    - 将 FlexMatch 添加到游戏客户端
- 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 Amazon GameLift 托管的游戏服务器
- 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 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：
    - 利用 FlexMatch 回填现有游戏

> **Note**
>
> 回填当前不能使用 用于游戏。

构建 Amazon GameLift FlexMatch 对战构建器

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 对战构建器如何处理它收到的对战请求的详细说明，请参阅 FlexMatch 对战流程。

主题

设计 FlexMatch 对战构建器

本主题提供有关如何设计适合您游戏的对战构建器的指导.

主题

配置一个对战构建器。

对战构建器至少需要具备三个元素：

- 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 规则集和FlexMatch 规则集示例。
- 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。
- 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。
- 将 FlexMatch 与 Amazon GameLift 托管托管资源配合使用时，游戏会话队列会找到最佳可用资源来托管对战的游戏会话，然后开始新的游戏会话。每个队列都配置了一系列位置和资源类型（包括竞价

型实例或按需型实例），用于确定游戏会话的放置位置。有关队列的更多信息，请参阅使用多位置队列。

为对战构建器选择一个区域

确定您希望进行对战活动的位置，并在该位置创建对战配置和规则集。Amazon GameLift 会为您的游戏对战请求维护票证池，并在其中对这些请求进行排序和评估请求中是否有可行的匹配。匹配完成后，Amazon GameLift 会发送对战详细信息，以便放置游戏会话。您可以在托管解决方案支持的任何位置运行匹配的游戏会话。

请参阅支持 FlexMatch 的 AWS 区域，了解可以创建 FlexMatch 资源的位置。

在为对战构建器选择 AWS 区域时，请考虑位置可能如何影响性能以及该位置可能如何优化玩家的对战体验。我们建议您遵循以下最佳实操：

- 将对战构建器放置在离您的玩家和发送 FlexMatch 对战请求的客户服务很近的地方。这种方法可以减少对对战请求工作流程的延迟影响，并提高其效率。
- 如果您的游戏吸引了全球观众，可以考虑在多个位置创建对战构建器，并将匹配请求发送给离玩家最近的对战构建器。除了提高效率外，这还会导致在地理位置上彼此靠近的玩家形成票证池，从而提高了对战构建器根据延迟要求匹配玩家的能力。
- 将 FlexMatch 与 Amazon GameLift 托管托管资源配合使用时，请将您的对战构建器及其使用的游戏会话队列放在同一位置。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。

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。如果您将 FlexMatch 与 Amazon GameLift 托管解决方案一起使用，则内置了许多功能。如果您使用 FlexMatch 作为独立的对战服务，则可能需要在系统中内置这些功能。

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。如果要求玩家接受，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，将会丢弃建议的对战游戏。对于所有玩家都接受对战游戏的票证，系统会将其状态返回到池中以继续处理。至少有一名玩家拒绝对战或未能回复的票证将进入失败状态，不再处理。玩家接受需要设置时间限制，您可以定义此限制；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

回填非常实用，可以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， 使用与匹配原始玩家相同的对战构建器和相同的过程来查找新玩家。您可以使用新对战的票证自定义回填票证的优先顺序，将回填票证放在排队的前面或末端。这意味着，当新玩家进入对战池时，他们被放置在现有游戏中的可能性大于或小于在新组建的游戏中。

无论您的游戏在托管 Amazon GameLift 托管上使用 FlexMatch 还是与其他托管解决方案搭配使用 FlexMatch，都可以进行手动回填。手动回填让您能够灵活地决定何时触发回填请求。例如，您可能不希望在游戏的某些阶段或仅存在某些条件时添加新玩家。

自动回填仅适用于使用托管 Amazon GameLift 托管的游戏。启用此功能后，如果游戏会话以开放的玩家插槽开始，Amazon GameLift 会开始自动为其生成回填请求。此功能允许您设置对战，以便以最少的玩家人数开始新游戏，然后在新玩家进入对战池时快速填满。在游戏会话生命周期内，您可以随时关闭自动回填功能。

游戏属性

对于使用 FlexMatch 和 Amazon GameLift 托管托管的游戏，您可以提供其他信息，以便在请求新的游戏会话时传递给游戏服务器。这可能是传递游戏模式配置的有用方法，这些配置是为正在创建的匹配类型启动游戏会话所必需的。由对战构建器创建的对战的所有游戏会话都将获得相同的游戏属性集。您可以通过创建不同的对战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。

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，规则集能够描述类似这样的对战游戏：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 AWS 控制台或AWS CLI。

主题

设计 FlexMatch 规则集

本主题介绍规则集的基本结构以及如何将它们用于最多 40 位玩家的对战游戏。在最基本的层面上，对战规则集有两个作用：安排对战游戏的团队结构和规模，以及告知对战构建器如何评估玩家以查找可能的最佳匹配。

但是您的对战规则集可以做得更多。例如，您可以：

- 优化游戏的对战算法。
- 设置最低玩家延迟要求以保护游戏质量。
- 随着时间的推移，逐渐放宽团队要求和对战规则，这样所有活跃的玩家都可以在需要时找到可以接受的对战。
- 使用玩家方聚合定义组对战请求的处理。
- 处理 40 名或更多玩家的大型对战。有关构建大型对战的更多信息，请参阅设计 大型对战规则集。

在建立对战规则集时，请考虑以下可选和必需任务：

- 描述规则集（必填）
- 自定义匹配算法
- 声明玩家属性
- 定义对战队伍
- 设置玩家匹配规则
- 允许要求随着时间的推移而放松

您可以使用 Amazon GameLift 控制台或 `CreateMatchmakingRuleSet` 操作来构建自己的规则集。

描述规则集（必填）

提供规则集的详细信息。

- 名称（可选）– 供您自己使用的描述性标签。此值与您在使用 Amazon GameLift 创建规则集时指定的规则集名称无关。
- ruleLanguageVersion – 这是用于创建 FlexMatch 规则的属性表达式语言的版本。值必须为 `1.0`。

自定义匹配算法

FlexMatch优化了大多数游戏的默认算法，以最少的等待时间让玩家进入可接受的对战。您可以自定义算法并调整游戏的对战。

以下是默认的 FlexMatch 对战算法：

1. FlexMatch 将所有未完成的对战票证和回填票证放入票池中。
2. FlexMatch 将池中的票证随机分组为一个或多个批次。随着票池的增大，FlexMatch 会形成额外的批次以保持最佳的批次大小。
3. FlexMatch 在每个批次中按年龄对票证进行排序。
4. FlexMatch 根据每批中最旧的票证建立匹配项。

要自定义匹配算法，请在您的规则集架构中添加一个 `algorithm` 组件。有关完整参考信息，请参阅FlexMatch 规则集架构。

使用以下可选自定义项来影响对战过程的不同阶段。

- 添加批处理前排序
- 根据 batchDistance 属性形成批次
- 优先考虑回填工单
- 偏爱带有扩展版的旧票证

添加批处理前排序

您可以在形成批次之前对票池进行排序。这种类型的自定义对于拥有大量票证池的游戏最为有效。预批次排序可以帮助加快对战过程并提高玩家在定义特征上的统一性。

使用算法属性 `batchingPreference` 定义批处理前的排序方法。默认设置为 `random`。

自定义批处理前排序的选项包括：

- 按玩家属性排序。提供玩家属性列表以对票证池进行预排序。

  要按玩家属性排序，batchingPreference 请设置为 sorted，然后在 sortByAttributes 中定义您的玩家属性列表。要使用属性，请先在规则集的 playerAttributes 组件中声明该属性。

  在以下示例中，FlexMatch 根据玩家的首选游戏地图，然后按玩家技能对票证池进行排序。生成的批次更有可能包含想要使用相同地图的技能相似的玩家。

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

- 按延迟排序。以最低的可用延迟创建匹配项，或者以可接受的延迟快速创建匹配项。此自定义对于形成超过 40 名玩家的大型对战的规则集非常有用。

  将算法属性 strategy 设置为 balanced。平衡策略限制了规则语句的可用类型。有关更多信息，请参阅 设计 大型对战规则集。

  FlexMatch 通过以下方式之一根据玩家报告的延迟数据对票证进行排序：

  - 延迟最低的位置。票证池按玩家报告最低延迟值的位置进行预先排序。然后，FlexMatch 在相同的位置以低延迟对票证进行批处理，从而创造更好的游戏体验。它还减少了每批票证的数量，因此对战可能需要更长的时间。要使用此自定义设置，请将 batchingPreference 设置为 fastestRegion，如以下示例所示。

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

  - 可接受的延迟很快就会匹配。票证池按玩家报告可接受延迟值的位置进行预先排序。这会形成更少的批次，包含更多的票证。每批票证越多，就能更快地找到可接受的匹配项。要使用此自定义设置，请将属性 batchingPreference 设置为 largestPopulation，如以下示例所示。

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

```
},
```

> **Note**
>
> 平衡策略的默认值为 `largestPopulation`。

优先考虑回填工单

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自定义 FlexMatch 处理对战票证的方式。请求类型可以是新的匹配请求或回填请求。默认情况下，FlexMatch 对两种类型的请求都一视同仁。

回填优先级会影响 FlexMatch 在对工单进行批处理后的处理方式。回填优先级要求规则集使用详尽的搜索策略。

FlexMatch 不会将多张回填票证匹配在一起。

要更改回填票证的优先级，请设置属性 `backfillPriority`。

- 先匹配回填票证。在创建新的匹配项之前，此选项会尝试匹配回填票证。这意味着新玩家加入现有游戏的几率更高。

  如果您的游戏使用自动回填功能，则最好使用此选项。自动回填通常用于游戏会话短、玩家周转率高的游戏。自动回填可以帮助这些游戏形成最低限度的可行匹配并开始对战，而FlexMatch则会搜索更多玩家来填补空缺位置。

  将 `backfillPriority` 设置为 `high`。

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

- 最后匹配回填票证。此选项会忽略回填票证，直到它评估所有其他票证。这意味着，当FlexMatch无法将新玩家匹配到新游戏时，它会将他们重新填充到现有游戏中。

  当您想使用回填作为最后机会让玩家进入游戏时，例如当没有足够的玩家来组建新的对战时，这个选项很有用。

  将 `backfillPriority` 设置为 `low`。

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

偏爱带有扩展版的旧票证

当对战难以完成时，扩展规则会放宽匹配标准。当部分完成的对战的票证达到一定年龄时，Amazon GameLift会适用扩展规则。票证的创建时间戳决定了 Amazon GameLift 何时应用规则；默认情况下，FlexMatch 会跟踪最近匹配的票证的时间戳。

要更改 FlexMatch 应用扩展规则的时间，请按以下方式设置 `expansionAgeSelection` 属性：

- 根据最新票证进行扩展。此选项根据添加到潜在匹配项中的最新票证来应用扩展规则。每当 FlexMatch 匹配新票证时，时钟都会重置。使用此选项，结果匹配的质量往往更高，但匹配时间更长；如果匹配请求需要太长时间才能匹配，则匹配请求可能会在完成之前超时。将 `expansionAgeSelection` 设置为 `newest`。`newest` 为默认值。
- 根据最旧的票证进行扩展。此选项根据潜在匹配中最旧的票证应用扩展规则。使用此选项，FlexMatch 可以更快地应用扩展，从而缩短最早匹配的玩家的等待时间，但会降低所有玩家的对战质量。将 `expansionAgeSelection` 设置为 `oldest`。

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

声明玩家属性

在本节中，列出要包含在对战请求中的个人玩家属性。您可能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规则时。
- 当您想通过匹配请求将玩家属性传递给游戏会话时。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（必需）–此值在规则集中必须唯一。

- 类型（必需）– 这是属性值的数据类型。有效数据类型为数字、字符串或字符串映射。
- 默认（可选）– 输入一个默认值，以便在对战请求未提供属性值时使用。如果未声明默认值且请求中不包含值，则 FlexMatch 无法满足请求。

定义对战队伍

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。

根据规则集如何定义团队规模将对战请求处理为小型对战或大型对战。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。您可以在规则和扩展中使用这个名字，在游戏会话中使用 FlexMatch 引用对战数据。
- maxPlayers（必需）– 指定可以分配给团队的玩家的最大数量。
- MinPlayers（必填）– 指定要分配给队伍的最小玩家人数。
- 数量（可选）– 使用此定义指定要组建的团队数量。当 FlexMatch 创建对战时，它会为这些团队提供所提供的名字和一个附加的数字。例如 `Red-Team1`、`Red-Team2` 和 `Red-Team3`。

FlexMatch 试图将队伍填满最大玩家人数，但确实会创建玩家人数较少的团队。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 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 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。

- 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。
- 类型（必需）– 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。有关有效的规则类型和属性的列表，请参阅 FlexMatch 规则语言。

- 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。在 FlexMatch 规则语言 中了解有关属性以及如何使用 FlexMatch 属性表达式语言的更多信息。

允许要求随着时间的推移而放松

当 FlexMatch 找不到匹配项时，扩展允许您随着时间的推移放宽规则标准。此功能可确保 FlexMatch 在无法实现完美匹配时提供最佳可用性。通过利用扩展放宽规则，您将逐渐扩大可以匹配的玩家池。

当未完成匹配中最新票证的年龄与扩展版等待时间相匹配时，扩展版就会开始。当 FlexMatch 向对战添加新票证时，扩展等待时间时钟可能会被重置。您可以在规则集的`algorithm`部分自定义扩展的开始方式。

以下是扩展版的示例，它会逐渐提高对战所需的最低技能等级。该规则集使用名为 SkillDelta 的距离规则语句来要求对战中的所有玩家彼此相距在 5 个技能等级以内。如果在十五秒钟内没有进行新的对战，则此扩展版将寻找技能等级差异为 10，然后在十秒钟后寻找 20 的差异。

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

如果此规则集由启用了自动回填的对战构建器使用，不要过快放宽玩家计数要求。新游戏会话需要几秒钟时间启动并开始自动回填。更好的方法是仅在您的游戏要开始自动回填后设置扩展等待时间。扩展时间因您的团实例集成而异，因此请进行测试以找到最适合您游戏的扩展策略。

设计 大型对战规则集

如果您的规则集创建的对战允许 41 到 200 名玩家，则您必须对您的规则集配置做出一些调整。这些调整优化了对战算法，使其可以建立可行的大型对战，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对战优先级进行了优化的标准解决方案取代了耗时的自定义规则。

以下是如何确定是否需要针对大型对战优化规则集：

1. 对于规则集中定义的每支队伍，获取 maxPlayer 的值，

2. 将所有 MaxPlayer 值相加。如果这些设置的总计超过 40，表示您有大型对战规则集。

要针对大型对战优化规则集，请按如下所述进行调整。参考架构，了解 大型对战的规则集架构 中的大型对战规则集和 参考：创建大型对战 中的规则集示例。

为大型对战自定义匹配算法

向规则集添加算法组件（如果尚不存在）。设置以下属性：

- `strategy`（必填）– 将 `strategy` 属性设置为“平衡”。此设置会触发 FlexMatch 进行额外的赛后检查，以根据 `balancedAttribute` 属性中定义的指定玩家属性找到最佳的团队平衡。平衡的策略取代了使用自定义规则来组建势均力敌的团队。
- `balancedAttribute`（必填）– 确定在对战中平衡团队时要使用的玩家属性。此属性必须具有数字数据类型（双精度或整数）。例如，如果您选择平衡玩家技能，FlexMatch 会尝试分配玩家，以便所有队伍的总技能水平尽可能平衡。请务必在规则集的玩家属性中声明平衡属性。
- `batchingPreference`（可选）– 选择您想在多大程度上强调为玩家打造尽可能低的延迟对战。此设置会影响在建立对战之前对对战票证的排序方式。选项包括：
  - 最大群体 FlexMatch 允许使用池中所有在至少一个共同位置具有可接受延迟值的票证进行匹配。因此，潜在的票证池往往很大，这使得更容易更快地填补对战。玩家可能被放置在延迟可接受但并不总是最佳延迟的游戏中。如果未设置 `batchingPreference` 属性，则这是 `strategy` 设置为“平衡”时的默认行为。
  - 最快的位置。FlexMatch 根据池中报告最低延迟值的位置对所有票证进行预排序。因此，往往会与报告相同位置低延迟的玩家进行对战。同时，每场对战的潜在票证池较小，这可能会增加填补对战所需的时间。此外，由于对延迟的优先级更高，因此对战中的玩家在平衡属性方面的差异可能更大。

以下示例将匹配算法配置为如下行为：(1) 对票池进行预排序，按票证具有可接受的延迟值的位置对票证进行分组；(2) 形成批量排序的票证进行匹配；(3) 批量创建带有票证的匹配并平衡队伍以平衡普通玩家技能。

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

声明玩家属性

至少，您必须声明在规则集算法中用作平衡属性的玩家属性。对战请求中，每个玩家都应包含此属性。您可以为玩家属性提供默认值，但是如果提供了玩家特定的值，则属性平衡效果最好。

定义团队

定义团队规模和结构的过程与小型对战相同，但 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。您可能想要更改响应中的最小团队规模。

在向团队分配玩家时使用以下规则。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，即使对战没有满员，一场对战中的团队也总是有几乎相等的玩家人数。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。然后，当空闲位置的数量在所有团队之间更均匀地分配，玩家开始被添加到更小的团队。

例如，假设您有一个由三支队伍组成的规则。红色团队和蓝色团队均设置为 `maxPlayers` = 10、`minPlayers` = 5。绿队设置为 `maxPlayers` = 3、`minPlayers` = 2。这是填充顺序：

1. 尚未有队伍到达 `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。这两个团队现在已达到 `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。绿队现已到达 `minPlayers`。
3. 在所有队伍都进入 `minPlayers` 的情况下，现在会根据空缺位置的数量分配更多玩家。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。所有队伍现在都有 1 个空缺位置。
4. 剩下的 3 个玩家名额（每个 1 个）分配给队伍，顺序不分先后。

为大型对战设定规则

大型对战的对战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。将 `latency` 规则类型与属性 `maxLatency` 一起使用。参阅 延迟规则 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
    }],
```

- 根据指定玩家属性中的接近程度对玩家进行批处理的规则。这与将平衡属性定义为大型对战算法的一部分不同，后者侧重于组建势均力敌的团队。此规则根据指定属性值（例如初学者或专家技能）的相似性对对战票证进行批处理，这往往会导致匹配在指定属性上紧密对齐的玩家。使用 batchDistance 规则类型，标识基于数字的属性，并指定允许的最大范围。参阅 Batch 距离规则 引用。以下是一个示例，要求对战的玩家彼此之间必须处于一个技能等级之内：

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

放宽大型对战要求

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补赛，请避免过快地放松您的队员人数。只有在游戏会话开始后，FlexMatch 才会开始生成回填请求，这种情况在创建匹配后的几秒钟内可能不会发生。在这段时间内， 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
        }]
    }]
```

教程：创建对战规则集

在为 Amazon GameLift FlexMatch 对战构建器创建对战规则集之前，我们建议您检查规则集语法。使用 Amazon GameLift 控制台或 AWS Command Line Interface（AWS CLI）创建规则集后，您无法对其进行更改。

注意您可以在 AWS 区域拥有的规则集最大数量有服务限额，因此，最好将不使用的规则集删除。

Console

创建规则集。

1. 通过以下网址打开 Amazon GameLift 控制台：https://console.aws.amazon.com/gamelift/。
2. 切换到您希望放置规则集的 AWS 区域。在要使用规则集的对战配置中，将规则集定义到同一区域中。
3. 在导航窗格中，选择 FlexMatch，对战规则集。
4. 在对战规则集页面上，选择创建规则集。
5. 在创建规则页面上，执行以下操作：
    a. 在规则集设置下，为名称输入一个唯一的描述性名称，您可以使用该名称在列表或事件和指标表中对其进行识别。
    b. 在规则集中，以 JSON 格式输入您的规则集。有关设计规则集的信息，请参见设计 FlexMatch 规则集。您也可以使用中的一个示例规则集FlexMatch 规则集示例。
    c. 选择验证 以验证您规则集的语法正确。规则集创建后您无法对其进行编辑，因此最好先对其进行验证。
    d. （可选）在标签下，添加标签以帮助您管理和跟踪 AWS 资源。
6. 选择创建。如果创建成功，对战构建器可以使用该规则集。

AWS CLI

创建规则集

打开命令行窗口并使用 create-matchmaking-rule-set 命令。

此示例创建设置单个团队的简单对战规则集。请确保创建规则集所在的 AWS 区域，与将引用该规则集的对战配置的区域相同。

```
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，Amazon GameLift 将返回一个 MatchmakingRuleSet 对象，其中包括您指定的设置。新规则集现在可由对战构建器使用。

Console

删除规则集

1. 通过以下网址打开 Amazon GameLift 控制台：https://console.aws.amazon.com/gamelift/。
2. 切换到您在其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. 在对战规则集页面上，选择要删除的规则集，然后选择删除。
5. 在删除规则集对话框中，选择删除以确认删除。

> **Note**
>
> 如果对战配置使用规则集，Amazon GameLift 会显示一条错误消息（无法删除规则集）。如果发生这种情况，请更改对战配置以使用其他规则集，然后重试。要了解一个规则集目前正被哪些对战配置使用，请单击该规则集名称，查看其详细信息页面。

AWS CLI

删除规则集

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

如果对战配置使用规则集，Amazon GameLift 会返回一条错误消息。如果发生这种情况，请更改对战配置以使用其他规则集，然后重试。要获取当前使用某个规则集的对战配置列表，请使用命令 describe-matchmaking-configurations 并指定该规则集名称。

此示例首先检查对战规则集的使用情况，然后删除该规则集。

```
aws gamelift describe-matchmaking-rule-sets \
    --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：

> **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。

主题

- 参考：创建两个势均力敌的玩家团队
- 参考：创建水平不对等的团队（猎人对战怪物）
- 参考：设置团队级要求和延迟限制
- 参考：使用显式排序以查找最佳对战游戏
- 参考：查找多个玩家属性的交集
- 参考：比较所有玩家的属性
- 参考：创建大型对战
- 参考：创建多团队大型对战
- 参考：创建与属性相似的玩家的大型对战
- 参考：使用复合规则创建与属性相似或选择相似的玩家的对战

- 参考：创建使用玩家屏蔽名单的规则

参考：创建两个势均力敌的玩家团队

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
        }]
    }]
}
```

参考：创建水平不对等的团队（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
        }]
    }]
}
```

参考：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。为评估 `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
        }]
    }]
}
```

参考：使用显式排序以查找最佳对战游戏

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。这些规则会为对战票证预排序，以根据特定的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
    }]
}
```

参考：查找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。

此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：玩家希望作为对手的角色的列表。
- 对战游戏规则：如果每个正在使用的角色都位于各个玩家的首选对手列表中，则潜在的对战游戏是可接受的。

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 – 使用 `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。

- 测量值 – 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` 属性表达式创建所有 preferredOpponents 属性值的字符串列表，这些属性值对于对战游戏中的每个玩家是通用的。
- 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. 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
        }]
    }]
}
```

参考：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。

此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“基于次序”)。
  - gameMap：玩家选择的游戏世界 (如果未提供，则默认设置为 1)。
  - character：玩家选择的角色（无默认值，表示玩家必须指定角色）。
- 对战游戏规则：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])"]
    }]
}
```

参考：创建大型对战

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。参阅设计 大型对战规则集了解更多信息。

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。
- 平衡条件：根据相似的技能级别选择玩家。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：在创建对战时按类似的平衡条件分组玩家。
- 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。参阅放宽大型对战要求了解更多信息。

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

参考：创建多团队大型对战

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。
- 平衡条件：根据怪物技能的数量选择玩家。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分组玩家。
- 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，这让它成为大型对战。
- 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。
- 下面介绍了 在有任何扩展生效之前如何针对此规则集填充对战：
  - 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。参阅放宽大型对战要求了解更多信息。

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

参考：创建与属性相似的玩家的大型对战

此示例说明如何使用 batchDistance 为两个团队的对战设置规则集。在示例中：

- 该 SimilarLeague 规则确保一场对战中的所有玩家与其他玩家具有 2 以内的 league。
- 该 SimilarSkill 规则确保一场对战中的所有玩家与其他玩家具有 10 以内的 skill。如果玩家等待 10 秒，则距离将扩大到 20 秒。如果玩家等待 20 秒，则距离将扩大到 40 秒。
- 该 SameMap 规则确保对战中的所有玩家都提出同样的 map。
- 该 SameMode 规则确保对战中的所有玩家都提出同样的 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
        }]
    }]
}
```

参考：使用复合规则创建与属性相似或选择相似的玩家的对战

此示例说明如何使用 compound 为两个团队的对战设置规则集。在示例中：

- 该 SimilarLeagueDistance 规则确保一场对战中的所有玩家与其他玩家具有 2 以内的 league。
- 该 SimilarSkillDistance 规则确保一场对战中的所有玩家与其他玩家具有 10 以内的 skill。如果玩家等待 10 秒，则距离将扩大到 20 秒。如果玩家等待 20 秒，则距离将扩大到 40 秒。
- 该 SameMapComparison 规则确保对战中的所有玩家都提出同样的 map。
- 该 SameModeComparison 规则确保对战中的所有玩家都提出同样的 mode。
- 如果满足以下任意条件，CompoundRuleMatchmaker 规则确保：
  - 一场对战中的玩家提出了同样的 map 和 mode。

- 对战中的玩家具有不相上下的 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
        }]
    }]
}
```

参考：创建使用玩家屏蔽名单的规则

此示例说明了一个规则集，该规则集允许玩家避免与某些其他玩家对战。玩家可以创建屏蔽名单，对战构建器在为对战选择玩家时对其进行评估。有关添加屏蔽名单或避免列表特征的更多指导，参阅 AWS 游戏博客。

此方案规定了以下说明：

- 创建两支由五名玩家组成的队伍。
- 传入玩家的屏蔽名单，这是玩家 ID 的列表（最多 100 个）。
- 将所有玩家与每位玩家的屏蔽名单进行比较，如果发现任何被封锁的玩家 ID，则拒绝提议的对战。

使用此规则集的说明：

- 在评估新玩家以加入拟议的对战（或填补现有对战中的位置）时，该玩家可能会因为以下任一原因而被拒绝：
  - 如果新玩家出现在任何已被选中参加对战的玩家的屏蔽名单上。
  - 如果新玩家的屏蔽名单上有已经被选中参加对战的玩家。
- 如图所示，该规则集禁止将玩家与其屏蔽名单上的任何玩家进行匹配。您可以通过添加规则扩展并增加该`maxCount`值来将此要求更改为首选项（也称为“避免列表）。

```
{
    "name": "Player Block List",
    "ruleLanguageVersion": "1.0",
    "teams": [{
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "red"
    }, {
        "maxPlayers": 5,
        "minPlayers": 5,
        "name": "blue"
    }],
    "playerAttributes": [{
        "name": "BlockList",
        "type": "string_list",
        "default": []
    }],
    "rules": [{
        "name": "PlayerIdNotInBlockList",
        "type": "collection",
        "operation": "reference_intersection_count",
        "measurements": "flatten(teams[*].players.attributes[BlockList])",
        "referenceValue": "flatten(teams[*].players[playerId])",
        "maxCount": 0
    }]
}
```

创建对战配置

要设置 Amazon GameLift FlexMatch 对战构建器来处理对战请求，请创建对战配置。使用 Amazon GameLift 控制台或 AWS Command Line Interface（AWS CLI）。有关创建对战构建器的更多信息，请参阅设计 FlexMatch 对战构建器。

主题

教程：为 Amazon GameLift 托管创建对战构建器

在创建对战配置之前，请创建一个规则集和一个 Amazon GameLift 游戏会话队列，以便与对战构建器一起使用。

Console

1. 在 Amazon GameLift 控制台的导航窗格中，选择对战配置。
2. 切换到要创建对战构建器的 AWS 区域。
3. 在对战配置页面上，选择创建对战配置。
4. 在定义配置详细信息页面的对战配置详细信息下，执行以下操作：
   a. 在姓名中，输入可以帮助您在列表和指标中识别匹配者的姓名。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）对于描述，添加有助于识别对战构建器的描述。
   c. 对于规则集，从列表中选择要与对战构建器一起使用的规则集。该列表包含在当前区域中已创建的所有规则集。
   d. 对于 FlexMatch 模式，请为 Amazon GameLift 托管托管资源选择已托管。此模式会提示 FlexMatch 将成功的匹配传递到指定的游戏会话队列。
   e. 对于 AWS区域，选择您配置要与对战构建器一起使用的游戏会话队列的区域。
   f. 对于队列，选择要用于该对战构建器的游戏会话队列。
5. 选择下一步。
6. 在配置设置页面的对战设置下，执行以下操作：
   a. 对于请求超时，键入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）。FlexMatch 会取消超过此时间的对战请求。
   b. 对于回填模式，请选择一种处理对战回填的模式。
      - 选择自动打开自动回填特征。
      - 要创建自己的回填请求管理或不使用回填特征，请选择手动。

c. （可选）对于额外玩家人数，请设置一场对战中要保持开放的玩家位置数量。将来玩家可以占用这些位置。

d. （可选）在对战接受选项下，在需要接受中，如果您想要求提议的对战中的每位玩家积极接受参与对战，请选择必填。如果您选择此选项，则在接受超时中，设置您希望对战构建器在取消对战之前等待玩家接受的时间（以秒为单位）。

7. （可选）在事件通知设置下，执行以下操作：

a. （可选）对于 SNS 主题，选择用于接收对战活动通知的 Amazon Simple Notification Service (Amazon SNS) 主题。如果您尚未设置，可以在以后通过编辑对战配置来添加此信息。有关更多信息，请参阅设置 FlexMatch 事件通知。

b. （可选）对于自定义事件数据，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。该数据包含在与对战构建器关联的每个事件中。

8. （可选）展开其他游戏数据，然后执行以下操作：

a. （可选）对于游戏会话数据，请输入您希望 FlexMatch 向使用此对战配置进行的匹配开始的新游戏会话提供的任何其他与游戏相关的信息。

b. （可选）对于游戏属性，添加包含有关新游戏会话信息的键值对属性。

9. （可选）在标签下，添加标签以帮助您管理和跟踪 AWS 资源。

10. 选择下一步。

11. 在查看和创建页面上，查看您的选择，然后选择创建。如果创建成功，则对战构建器会立即准备好接受对战请求。

AWS CLI

要使用 AWS 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。

此示例命令创建了一个新的对战配置，该配置需要玩家接受并启用自动回填功能。它还为 FlexMatch保留了两个玩家位置，以便以后添加玩家，并提供一些游戏会话数据。

```
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，Amazon GameLift 会返回一个 MatchmakingConfiguration 对象，其中包含为对战构建器请求的设置。新对战构建器已准备好接受对战请求。

教程：为独立版 FlexMatch 创建对战构建器

在创建对战配置之前，必须创建要与对战构建器一起使用的规则集。

Console

1. 通过以下网址打开 Amazon GameLift 控制台：https://console.aws.amazon.com/gamelift/home。
2. 切换到要创建对战构建器的 AWS 区域。有关支持 FlexMatch 对战配置的区域的列表，请参阅为对战构建器选择一个区域。
3. 在导航窗格中，选择 FlexMatch、对战配置。
4. 在对战配置页面上，选择创建对战配置。
5. 在定义配置详细信息页面的对战配置详细信息下，执行以下操作：
   a. 在姓名中，输入可以帮助您在列表和指标中识别匹配者的姓名。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）对于描述，添加有助于识别对战构建器的描述。
   c. 对于规则集，从列表中选择要与对战构建器一起使用的规则集。该列表包含在当前区域中已创建的所有规则集。
   d. 对于 FlexMatch 模式，请选择独立。这表明您有一个自定义机制，可以在 Amazon GameLift 之外的托管解决方案上启动新的游戏会话。
6. 选择下一步。
7. 在配置设置页面的对战设置下，执行以下操作：

a. 对于请求超时，键入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终止。

b. （可选）在对战接受选项下，在需要接受中，如果您想要求提议的对战中的每位玩家积极接受参与对战，请选择必填。如果您选择此选项，则在接受超时中，设置您希望对战构建器在取消对战之前等待玩家接受的时间（以秒为单位）。

8. （可选）在事件通知设置下，执行以下操作：

a. （可选）对于 SNS 主题，请选择一个 Amazon SNS 主题以接收对战活动通知。如果您尚未设置，可以在以后通过编辑对战配置来添加此信息。有关更多信息，请参阅设置 FlexMatch 事件通知。

b. （可选）对于自定义事件数据，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。该数据包含在与对战构建器关联的每个事件中。

9. （可选）在标签下，添加标签以帮助您管理和跟踪 AWS 资源。

10. 选择下一步。

11. 在查看和创建页面上，查看您的选择，然后选择创建。如果创建成功，则对战构建器会立即准备好接受对战请求。

AWS CLI

要使用 AWS 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对战构建器创建新的对战配置。

```
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，Amazon GameLift 会返回一个 MatchmakingConfiguration 对象，其中包含为对战构建器请求的设置。新对战构建器已准备好接受对战请求。

教程：编辑对战配置

要编辑对战配置，请从导航栏中选择对战配置，然后选择要编辑的配置。您可以更新现有配置中除名称之外的任何字段。

更新配置规则集时，如果存在有效的对战票，则新规则集可能不兼容，原因如下：

- 新的或不同的队伍名称或队伍数量
- 玩家属性
- 对现有玩家属性类型的更改

要对规则集进行任何更改，请使用更新的规则集创建新的对战配置。

设置 FlexMatch 事件通知

您可以使用事件通知来跟踪个人对战请求的状态。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。

- 让您的对战构建器将事件通知发布到 Amazon Simple Notification Service 主题。
- 使用自动发布的 Amazon EventBridge 赛事及其工具套件来管理事件。

有关 Amazon GameLift 发出的 FlexMatch 事件的列表，请参阅FlexMatch 对战活动。

主题

- 教程：设置 EventBridge 事件
- 教程：设置 Amazon SNS 主题
- 教程：使用服务器端加密设置 SNS 主题
- 教程：将主题订阅配置为调用 Lambda 函数

教程：设置 EventBridge 事件

Amazon GameLift 会将所有对战的事件发布到 Amazon EventBridge。使用 EventBridge，您可以设置规则，将对战的事件传送到目标进行处理。例如，您可以设置一个规则，将事

件“PotentialMatchCreated”路由到处理玩家接受情况的 AWS Lambda 函数。有关更多信息，请参阅什么是 Amazon EventBridge？

> **Note**
>
> 在配置对战构建器时，如果您想同时使用 EventBridge 和 Amazon SNS，请将通知目标字段保留为空或引用 SNS 主题。

教程：设置 Amazon SNS 主题

您可以让 Amazon GameLift 将 FlexMatch 对战构建器生成的所有事件发布到 Amazon SNS 主题中。

为 Amazon GameLift 事件通知创建 SNS 主题

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《Amazon Simple Notification Service 开发人员指南》中的创建主题AWS Management Console。
5. 在主题的详细信息页面上，选择编辑。
6. （可选）在主题的编辑页面上，展开访问策略，然后将以下 AWS Identity and Access Management (IAM) 策略声明中的粗体语法添加到现有策略的末尾。(为清晰起见显示了整个策略。) 请务必将 Amazon 资源名称 (ARN) 详细信息用于您自己的 SNS 主题和 Amazon GameLift 对战配置。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
```

```
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "your_account"
        }
      }
    },
    {
      "Sid": "__console_pub_0",
      "Effect": "Allow",
      "Principal": {
        "Service": "gamelift.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:your_region:your_account:your_topic_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
 "arn:aws:gamelift:your_region:your_account:matchmakingconfiguration/
your_configuration_name"
        }
      }
    }
  ]
}
```

7. 选择 Save changes（保存更改）。

教程：使用服务器端加密设置 SNS 主题

您可以使用服务器端加密 (SSE)，采用加密主题的方式传输敏感数据。SSE 使用 AWS Key Management Service (AWS KMS) 中托管的密钥保护 Amazon SNS 主题中消息的内容。有关 Amazon S3 如何执行加密的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南中的使用服务器端加密保护数据。

要使用服务器端加密设置 SNS 主题，请查看下面的主题：

- 《AWS Key Management Service 开发人员指南》中的创建密钥。
- 将 S imple Notification Service 开发人员指南中的主题启用 SSE

创建 KMS 密钥时，请使用以下 KMS 密钥策略：

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

教程：将主题订阅配置为调用 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，将通知目标字段设置为 SNS 主题 ARN。

以下AWS CloudFormation模板将订阅名为的 SNS 主题配置为调用名MyFlexMatchEventTopic为的 Lambda 函数。FlexMatchEventHandlerLambdaFunction该模板创建了一个 IAM 权限策略，允许 Amazon GameLift 写入 SNS 主题。然后，模板将调用 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
 AWS managed key
    Subscription:
      - Endpoint: !GetAtt FlexMatchEventHandlerLambdaFunction.Arn
        Protocol: lambda
    TopicName: MyFlexMatchEventTopic

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

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

为 FlexMatch 准备游戏

使用 Amazon GameLift FlexMatch 为您的游戏添加玩家对战功能。 可与托管 解决方案一起用于自定义游戏服务器和 。

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 规则集示例中有关自定义对战的更多详细信息。

此外，FlexMatch 基于队列功能进行构建。对战游戏组成之后， 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 实例集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 构建 Amazon GameLift FlexMatch 对战构建器。有关 FlexMatch 如何运行的更多信息，请参阅Amazon GameLift FlexMatch 的工作原理。

将 FlexMatch 添加到游戏客户端

本主题介绍如何为您的游戏客户端添加 对战支持。无论您是在 Amazon GameLift 托管托管资源上使用 FlexMatch，还是与其他托管解决方案一起使用 FlexMatch，过程基本相同。要了解有关 以及如何为游戏设置自定义对战构建器的更多信息，请参阅以下主题：

- 教程：将 FlexMatch 与 Amazon GameLift 托管集成
- Amazon GameLift FlexMatch 的工作原理
- 构建 Amazon GameLift FlexMatch 对战构建器
- FlexMatch 规则集示例

要为您的游戏启用对战回填，请添加以下功能：

- 请求一个或多个玩家的对战。
- 跟踪对战请求的状态。
- 要求玩家接受建议的对战游戏。
- 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。

请求玩家对战

我们强烈建议您的游戏客户端通过客户端游戏服务而不是直接发起对战请求。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 Amazon GameLift API。您的客户端服务使用 Amazon GameLift API 中的功能，它是AWS 软件开发工具包的一部分。请参阅适用于客户端服务的 Amazon GameLift 软件开发工具包，了解有关 AWS 软件开发工具包的更多信息并下载最新版本。将此软件开发工具包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 利用 FlexMatch 回填现有游戏 中了解有关管理对战回填的更多信息。

请求玩家对战

将代码添加到您的客户端服务来创建和管理 对战构建器的对战请求。对于使用 FlexMatch 和 Amazon GameLift 托管托管的游戏，以及使用 FlexMatch 作为独立解决方案的游戏，申请 FlexMatch 对战的过程是相同的。

创建对战请求：

- 调用 Amazon GameLift API StartMatchmaking。每个请求都必须包含以下信息。

  对战构建器

  指定要用于请求的对战配置的名称。要用于请求的对战配置的名称。 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 FlexMatch 对战构建器 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。当一个请求中有多个玩家时， 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。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 规则集和FlexMatch 规则集示例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，您可以通过调用包含请求票证 ID 的 DescribeMatchmaking 查看请求详细信息。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：

- 您随时可以通过调用包含请求票证 ID 的 StopMatchmaking 取消对战请求。

对战事件

设置通知，以跟踪 为对战过程发出的事件。您可以直接设置通知，也可以通过创建 SNS 主题或使用来设置通知。有关设置通知的更多信息，请参阅设置 FlexMatch 事件通知。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。

通过调用包含请求票证 ID 的 DescribeMatchmaking 检索对战请求票证，包括当前状态。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 FlexMatch 和 Amazon GameLift 托管的主机的游戏，以及使用 FlexMatch 作为独立解决方案的游戏，管理玩家接受度的过程是相同的。

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 DescribeMatchmaking 来检索对战游戏详细信息。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 FlexMatch 报告玩家响应。通过调用包含接受或拒绝的 AcceptMatch 报告玩家响应。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。接受对战的玩家的票证将自动退还到票证池中。未接受对战的玩家的票证将变为“失败”状态，不再受理。对于有多名玩家的票证，如果票证中有任何玩家不接受对战，则整张票证失效。

连接到对战游戏

将代码添加到您的客户端服务以处理已成功完成的对战（状态 COMPLETED 或事件 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。

对于使用 Amazon GameLift 托管托管资源的游戏，成功完成对战请求后，游戏会话连接信息将添加到对战票证中。通过调用 DescribeMatchmaking 检索已完成的对战票证。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。您的游戏客户端将使用此信息直接连接到托管对战的游戏会话。已对战的游戏会话的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）。

对于使用其他托管解决方案（包括 Amazon GameLift FleetIQ）的游戏，您必须建立一种机制，使对战玩家能够连接到相应的游戏会话。

示例 StartMatchmaking 请求

这些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 Amazon GameLift 托管的游戏服务器

本主题介绍如何向使用 Amazon GameLift 托管托管的自定义游戏服务器添加 FlexMatch 对战支持。要了解有关如何向游戏添加 的更多信息，请参阅以下主题：

- Amazon GameLift FlexMatch 的工作原理
- 教程：将 FlexMatch 与 Amazon GameLift 托管集成

本主题中的信息假定您已将 服务器开发工具包成功集成到您的游戏服务器项目中，如中所述。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。要请求新的游戏会话， 将向您的游戏服务器发送一个 请求，并在其中包含游戏会话对象（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，请参阅开始游戏会话。

   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 中了解更多信息。

2. 处理玩家连接。当连接到已经匹配的游戏时，游戏客户端将引用玩家 ID 和玩家会话 ID（请参阅 ）。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将调用服务器 API `RemovePlayerSession()` 来报告已退出的玩家（请参阅 ）。如果您使用 回填填充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 回填的更多信息。

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 利用 FlexMatch 回填现有游戏 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，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 创建的游戏会话， 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，此 ARN 将标识配置名称、AWS 账户和区域。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}}
   }
  }]
 }]
}
```

利用 FlexMatch 回填现有游戏

匹配回填使用 机制为现有的已匹配游戏会话寻找新玩家。尽管您可以始终向任何游戏添加玩家（请参阅将玩家加入游戏会话），但是对战回填可确保新玩家满足与当前玩家相同的匹配条件。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 对战流程中了解有关对战回填的更多信息。

> **Note**
>
> FlexMatch 回填当前不能使用实时服务器用于游戏。

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。

打开自动回填

使用自动匹配回填时，每当游戏会话开始时有一个或多个玩家位置未满，Amazon GameLift 都将自动触发回填请求。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 最初会找到六名玩家，组成对战，然后开始新的游戏会话。使用自动回填时，新游戏会话可以立即要求增加四名玩家。根据游戏的性质，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，它将用于使用该对战构建器创建的所有匹配游戏会话。游戏服务器上启动游戏会话后，Amazon GameLift 就会开始为非完整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
2. 开启回填优先级。自定义您的对战流程，以便在创建新匹配项之前优先填写回填请求。在对战规则集中，添加算法组件并将回填优先级设置为“高”。有关更多详细信息，请参阅自定义匹配算法。
3. 使用新的对战数据更新现有游戏会话。Amazon GameLift 使用服务器软件开发工具包回调函数 `onUpdateGameSession` 使用对战信息更新您的游戏服务器（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 在游戏服务器上更新对战数据 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，请将代码添加到您的游戏客户端或游戏服务器来发起 Amazon 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（从游戏服务器）

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，请参阅教程：将 FlexMatch 与 Amazon GameLift 托管集成。

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 在游戏服务器上更新对战数据。

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构建器。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 (GameSession) 中，该对象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 GetGameSessionId() 来获取游戏会话 ARN。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 – 包含您正在回填的游戏会话中所有当前玩家 (Player) 的玩家信息。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。如果您不使用回填，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 (GameSession)、`MatchmakerData` 属性中提取玩家数据（请参阅使用对战构建器数据）。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。Amazon GameLift 使用服务器软件开发工具包回调函数 onUpdateGameSession 针对回填请求的状态更新您的游戏服务器（请参阅初始化服务器进程)。添加代码以在在游戏服务器上更新对战数据中处理状态消息（以及由于回填请求成功更新的游戏会话对象）。

   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（从客户端服务）

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，请参阅教程：将 FlexMatch 与 Amazon GameLift 托管集成。

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 在游戏服务器上更新对战数据。

与其他客户端功能一样，客户端游戏服务将结合使用 AWS 软件开发工具包与 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 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 (请参阅请求玩家对战)，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 – 识别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 – 包含您正在回填的游戏会话中所有当前玩家 (Player) 的玩家信息。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。如果您不使用回填，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)。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 (请参阅对战事件)。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。请在在游戏服务器上更新对战数据上查看更多信息。

在游戏服务器上更新对战数据

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 完成匹配回填请求时（成功与否），它将使用回调函数 onUpdateGameSession 调用您的游戏服务器。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 StopMatchmaking（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. 在您的游戏服务器对服务器软件开发工具包操作 ProcessReady()的调用中，添加 onUpdateGameSession 回调方法名称作为过程参数。

使用 FlexMatch 实现高安全性

AWS 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 AWS 的客户，您也可以从这些数据中心和网络架构受益。

安全性是 AWS 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 的安全性。

Amazon GameLift FlexMatch 参考

本部分包含与 进行对战的参考文档。

主题

- Amazon GameLift FlexMatch API 参考 (SDK) AWS
- FlexMatch 规则语言
- FlexMatch 对战活动

Amazon GameLift FlexMatch API 参考 (SDK) AWS

本主题提供了基于任务的 Amazon GameLift FlexMatch API 操作列表。 Amazon GameLift FlexMatch 服务 API 已打包AWS到命名空间中的软件开发工具包中。`aws.gamelift`下载 AWS 软件开发工具包或查看 Amazon GameLift API 参考文档。

Amazon GameLift FlexMatch 提供对战服务，用于使用 Amazon GameLift 托管解决方案托管的游戏（包括自定义游戏服务器或实时服务器的托管托管，以及使用 Amazon GameLift FleetIQ 在 Amazon EC2 上托管）以及其他托管系统，例如点对点、本地或云计算基元。有关其他 Amazon GameLift 托管选项的更多信息，请参阅 Amazon GameLift 开发人员指南。

主题

- 设置对战规则和流程
- 为一个或多个玩家申请对战
- 可用编程语言
- Amazon GameLift FlexMatch 发行说明和软件开发工具包 版本
- Amazon GameLift 开发人员资源

设置对战规则和流程

调用这些操作来创建 FlexMatch 对战构建器，为您的游戏配置对战流程，并定义一组用于创建对战和队伍的自定义规则。

对战配置

- CreateMatchmakingConfiguration 创建对战配置，其中包含构建玩家组和加入新游戏会话的说明。使用 Amazon GameLift 进行托管时，还要指定如何为对战创建新的游戏会话。

- DescribeMatchmakingConfigurations 检索 区域中定义的对战配置。
- UpdateMatchmakingConfiguration 更改对战配置队列的设置。
- 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 区域中定义的对战规则集。
- ValidateMatchmakingRuleSet 验证一组对战规则的语法。
- DeleteMatchmakingRuleSet 从区域中删除对战规则集。

为一个或多个玩家申请对战

从游戏客户端服务调用这些操作来管理玩家对战请求。

- StartMatchmaking 为一个玩家或想要一起玩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对战，在玩家接受推荐的对战时进行注册。
- 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

支持 Amazon GameLift 的 AWS 软件开发工具包有以下语言版本。有关开发环境支持的信息，请参阅每种语言的文档。

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

Amazon GameLift FlexMatch 发行说明和软件开发工具包 版本

发行说明提供与服务相关的新功能、更新和修复的详细信息。本页还包括 Amazon GameLift软件开发工具包 版本历史记录。

Amazon GameLift 开发人员资源

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

FlexMatch 规则语言

本节中的参考主题描述了用于构建对战规则以供 Amazon GameLift FlexMatch 使用的语法和语义。有关编写对战规则和规则集的详细帮助，请参阅构建 FlexMatch 规则集。

主题

- FlexMatch 规则集架构
- FlexMatch 规则集属性定义
- FlexMatch 规则类型
- FlexMatch 属性表达式

FlexMatch 规则集架构

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说明，请参阅FlexMatch 规则集属性定义。

小型对战的规则集架构

以下架构记录了规则集的所有可能属性和允许的值，该规则集用于建立最多 40 名玩家的对战。

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

```
        "batchingPreference": <"random", "sorted">,
        "sortByAttributes": [ "string" ],
        "expansionAgeSelection": <"newest", "oldest">,
        "backfillPriority": <"normal", "low", "high">
    },
    "teams": [{
        "name": "string",
        "maxPlayers": number,
        "minPlayers": number,
        "quantity": integer
    }],
    "rules": [{
        "type": "distance",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "maxDistance": number,
        "minDistance": number,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "comparison",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "operation": <"<", "<=", "=", "!=", ">", ">=">,
        "partyAggregation": <"avg", "min", "max">
        },{
        "type": "collection",
        "name": "string",
        "description": "string",
        "measurements": "string",
        "referenceValue": number,
        "operation": <"intersection", "contains", "reference_intersection_count">,
        "maxCount": number,
        "minCount": number,
        "partyAggregation": <"union", "intersection">
        },{
        "type": "latency",
        "name": "string",
        "description": "string",
        "maxLatency": number,
        "maxDistance": number,
```

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

大型对战的规则集架构

以下架构记录了规则集的所有可能属性和允许的值，该规则集用于构建超过 40 名玩家的对战。如果规则集中定义的所有团队的 值超过 40， 将按照大型对战指南处理使用此规则集的所有请求。

```
{
```

```
    "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。

`name`

规则集的描述性标签。此值与分配给 Amazon GameL ift MatchMakingRuleSet 资源的名称无关。此值包含在描述已完成匹配的对战数据中，但任何 Amazon GameLift 流程均未使用该值。

允许的值：字符串：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： 或

必填？ 是

`playerAttributes`

包含在对战请求中并用于对战过程的玩家数据的集合。您还可以在此处声明属性，以便将玩家数据包含在传递给游戏服务器的对战数据中，即使对战过程中未使用这些数据也是如此。

必填？ 否

`name`

玩家属性的唯一名称，供对战构建器使用。此名称必须与对战请求中引用的玩家属性名称相匹配。

允许的值：字符串：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”、“字符串列表”、“string_number_map”

必填？ 是

`default`

当对战请求未提供给玩家的默认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

用于自定义对战过程的可选配置设置。

必填？ 否

strategy

构建匹配时使用的方法。如果未设置此属性，则默认行为为“exhaustiveSearch”。

允许的值：

- “EximativeSearch” – 标准匹配方法。FlexMatch 根据一组自定义匹配规则评估池中的其他票证，围绕一批中最旧的票证形成匹配。此策略适用于40名或更少玩家的对战。使用此策略时，batchingPreference 应设置为“随机”或“已排序”。
- “平衡” – 经过优化的方法，可快速形成大型匹配项。此策略仅适用于 41 到 200 名玩家的对战。它通过对票证池进行预排序，建立潜在的对战并将玩家分配给团队，然后使用指定的玩家属性平衡对战中的每支团队来形成对战。例如，此策略可以用来平衡一场对战中所有团队的平均技能水平。使用此策略时，balancedAttribute 必须将其设置，batchingPreference 应设置为“最大人群”或“最快区域”。此策略无法识别大多数自定义规则类型。

必填？ 是

batchingPreference

在对票证进行分组以进行对战之前使用的预排序方法。预先对票证池进行排序会根据特定特征将票证批量组合在一起，这往往会提高最后一场对战中玩家的统一性。

允许的值：

- “随机”– 仅在 strategy =“EximativeSearch”时有效。不进行预排序；池中的票证是随机批量排序的。这是详尽搜索策略的默认行为。
- “已排序”– 仅在 strategy =“EximativeSearch”时有效。票证池是根据中列出的玩家属性进行预先排序的。sortbyAttributes
- “最大人群”– 仅在 strategy =“平衡”时有效。票证池是按玩家报告延迟水平可接受的区域进行预先排序的。这是默认的平衡策略的默认行为。
- “最快区域”– 仅在 strategy =“平衡”时有效。票证池按玩家报告最低延迟水平的地区进行预先排序。由此产生的对战需要更长的时间才能完成，但是所有玩家的延迟往往很低。

必填？ 是

balancedAttribute

使用平衡策略进行大型对战时要使用的玩家属性的名称。

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

必填？ 是的，如果 strategy =“平衡”。

sortByAttributes

在批处理之前对票证池进行预排序时要使用的玩家属性列表。此属性仅在使用详尽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 对字母和数值使用标准排序惯例。

允许的值：中声明的任何属性playerAttributes。

必填？ 是的，如果 batchingPreference =“已排序”。

backfillPriority

用于匹配回填票证的优先级排序方法。此属性确定 FlexMatch 何时批量处理回填票证。它仅在使用详尽搜索策略进行预排序时使用。如果未设置此属性，则默认行为为“正常”。

允许的值：

- “正常” – 在形成匹配项时，不考虑票证的请求类型（回填或新匹配项）。
- “高” – 工单批次按请求类型（然后按年龄）排序，FlexMatch 会先尝试匹配回填票证。
- “低” – 工单批次按请求类型（然后按年龄）排序，FlexMatch 会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。如果对战在一定时间后仍未完成，则扩展版用于放宽对战要求。等待时间是根据已部分填满的对战中已有票证的使用年限计算的。如果未设置此属性，则默认行为为“最新”。

允许的值：

- “最新” – 扩展等待时间是根据部分完成的对战中带有最新创建时间戳的票证计算得出的。扩展触发的速度往往更慢，因为一张较新的票证可以重新启动等待时间。
- “最旧” – 扩展等待时间是根据对战中创建时间戳最早的票据计算得出的。扩展往往会更快地触发。

必填？ 否

teams

一场对战中团队的配置。为每支队伍提供团队名称和规模范围。规则集必须定义至少一支团队。

name

HSM 的唯一名称。可以在规则和扩展中提及团队名称。成功对战后，玩家将在对战数据中按队名进行分配。

允许的值：字符串：字符串

必填？ 是

maxPlayers

maxPlayers（必需）指定可以分配给团队的玩家的最大数量。

允许的值：数值：数字

必填？ 是

minPlayers

在对战可行之前，必须分配到团队的最低玩家人数。

允许的值：数值：数字

必填？ 是

quantity

在一场对战中要创建的此类团队的数量。数量大于 1 的队伍会附加一个数字（“Red_1”、“Red_2” 等）。如果未设置该属性，默认值为 7。

允许的值：数值：数字

必填？ 否

rules

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。

必填？ 否

name

HSM 的唯一名称。规则集中的所有规则都必须具有唯一的名称。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。

允许的值：字符串：字符串

必填？ 是

description

规则的文字描述。此信息可用于确定规则的用途。它不用于对战过程。

允许的值：字符串：字符串

必填？ 否

type

规则声明的类型。每种规则类型都有其他必须设置的属性。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。

允许的值：

- “absoluteSort”– 使用显式排序方法进行排序，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，对批次中的票证进行排序。
- “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。
- “复合” – 使用规则集中其他规则的逻辑组合定义复合对战规则。仅支持玩家人数不超过 40 人的对战。
- “距离”-测量数字值之间的距离。
- “batchDistance”-测量属性值之间的差异并将其用于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旧的票证的比较情况对批次中的票证进行排序。
- “延迟” – 评估为对战请求报告的区域延迟数据。

必填？ 是

expansions

当对战无法完成时，随着时间的推移放宽对战要求的规则。将扩展设置为一系列逐步应用的步骤，以便更容易找到匹配项。默认情况下，FlexMatch 根据添加到对战中的最新票证的年龄来计算等待时间。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应比上一步长。例如，要安排一系列逐步扩展，可以使用 30 秒、40 秒和 50 秒的等待时间。等待时间不能超过匹配请求允许的最长时间，该时间在对战配置中设置。

必填？ 否

target

要放松的规则集元素。您可以放宽团队规模属性或任何规则语句属性。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`。在名为“minSkill”的比较规则语句中更改最低技能要求:`rules[minSkill].referenceValue`.

必填？ 是

steps

waitTimeSeconds

在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 距离规则

```
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小匹配项。有两种类型的批量距离规则：

- 比较数值属性值。例如，某项距离规则可能要求所有玩家必须在相邻的两个级别内。对于这种类型，请定义所有票证之间的最`batchAttribute`大距离。
- 比较字符串属性值。例如，这种类型的批量距离规则可能要求一场对战中的所有玩家都请求相同的游戏模式。对于此类型，请定义 FlexMatch 用于形成批次的`batchAttribute`值。

距离规则属性

- **batchAttribute**– 用于形成批次的玩家属性值。
- **maxDistance**– 成功匹配的最大距离值。用于比较数值属性。

- **partyAggregation** – 该值决定 FlexMatch 如何处理多名玩家（多方）的票证。有效的选项包括票证玩家的最小值（min）、最大值（max）和平均（avg）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@@ 参考值进行比较。例如，这种类型的比较规则可能要求匹配的玩家具有一定的技能等级或更高。对于此类型，请指定玩家属性、参考值和比较操作。
- 比较不同玩家。例如，这种类型的比较规则可能要求对战中的所有玩家使用不同的角色。对于这种类型，请指定玩家属性以及等于（=）或不等于（!=）的比较操作。不要指定参考值。

> **Note**
>
> Batch 距离规则可以更有效地比较玩家属性。为了减少对战延迟，请尽可能使用批量距离规则。

比较规则属性

- 要比较的玩家属性值。
- **referenceValue**– 与预期匹配的测量值进行比较的值。
- **operation**– 决定如何将测量值与参考值进行比较的值。有效操作包括：。
- **partyAggregation** – 该值决定 FlexMatch 如何处理多名玩家（多方）的票证。有效的选项包括票证玩家的最小值（min）、最大值（max）和平均（avg）值。默认为 avg。

距离规则 ()

```
distance
```

距离规则用于度量两个数字值之间的差值，例如技能级别之间的差距。例如，距离规则可能要求所有玩家玩游戏至少 30 小时。

> **Note**
>
> Batch 距离规则可以更有效地比较玩家属性。为了减少对战延迟，请尽可能使用批量距离规则。

距离规则属性

- **measurements**– 要测量距离的玩家属性值。这必须是带有数值的属性。
- **referenceValue** &endash; 用于对照潜在对战游戏度量距离的数字值。
- / 为了成功进行对战游戏而允许的最大或最小距离值。
- **partyAggregation** – 该值决定 FlexMatch 如何处理多名玩家（多方）的票证。有效的选项包括票证玩家的最小值（min）、最大值（max）和平均（avg）值。默认为 avg。

收集规则 ()

```
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/或采用集合格式 (字符串列表) 的一个玩家属性。例如，收集规则可能会考虑队伍中玩家选择的角色。然后，该规则可能会要求队伍至少拥有某个角色中的一个。

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。
- 值或值集合，用于对照潜在对战游戏评估测量值。
- **operation**– 决定如何比较一组测量值的值。有效选项包含以下内容：
  - intersection– 此操作用于衡量所有玩家收藏中相同值的数量。有关使用交叉操作的规则的示例，请参见参考：使用显式排序以查找最佳对战游戏。
  - contains 计量包含特定参考值的玩家属性集合的数量。有关使用“包含”操作的规则的示例，请参见参考：设置团队级要求和延迟限制。
  - 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相匹配的物品数量。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，请参阅参考：查找多个玩家属性的交集。
- / 为了成功进行对战游戏而允许的最大或最小计数值。
- **partyAggregation** – 该值决定 FlexMatch 如何处理多名玩家（多方）的票证。对于此值，您可以使用union合并队伍中所有玩家的玩家属性。或者，您可以intersection使用队伍的共同玩家属性。默认为 union。

复合规则

```
compound
```

复合规则使用逻辑语句来形成40名或更少玩家的对战。可以在单个规则集中使用多个复合规则。使用多个复合规则时，所有复合规则都必须为 true 才能形成匹配项。

您不能使用扩展规则扩展复合规则，但可以扩展基础规则或支持规则。

复合规则属性

- **statement**– 用于组合单个规则以形成复合规则的逻辑。您在此属性中指定的规则必须是在规则集的早期定义的。不能在复合batchDistance规则中使用规则。

  此属性支持以下逻辑运算符：
  - and– 如果提供的两个参数均为真，则表达式为真。
  - or– 如果提供的两个参数中的任何一个为真，则表达式为真。
  - not– 反转表达式中参数的结果。
  - xor– 如果只有一个参数为真，则表达式为真。

Example 示例

以下示例根据玩家选择的游戏模式匹配不同技能等级的玩家。

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

延迟规则

```
latency
```

延迟规则衡量每个位置的玩家延迟。延迟规则会忽略任何延迟时间高于最大值的位置。玩家在至少一个位置的延迟值必须低于最大值，延迟规则才能接受。通过指定maxLatency属性，您可以将此规则类型用于大型匹配项。

延迟规则属性

- **maxLatency**-某个位置的最大可接受延迟值。如果工单没有延迟低于最大值的位置，则该票证不符合延迟规则。
- **maxDistance**– 每张票证的延迟时间与距离参考值之间的最大值。
- **distanceReference**– 用于比较工单延迟的延迟值。距离参考值的最大距离之内的票证将成功匹配。有效选项为玩家延迟的最小值（min）或平均值（avg）。
- **partyAggregation** – 该值决定 FlexMatch 如何处理多名玩家（多方）的票证。有效的选项包括票证玩家的最小值（min）、最大值（max）和平均（avg）值。默认为 avg。

> **Note**
>
> 队列可以将游戏会话放置在与延迟规则不匹配的区域。有关队列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 ()

```
absoluteSort
```

绝对排序规则根据指定的玩家属性对一批对战票证进行排序，与添加到该批次的第一张票据进行比较。

绝对排序规则属性

- **sortDirection** – 对对战票证进行排序的顺序。有效选项包括：
- **sortAttribute**– 用于对票证进行排序的玩家属性。
- **mapKey**– 如果是地图，则用于对玩家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– 值最高的密钥是第一个。
- **partyAggregation** – 该值决定 FlexMatch 如何处理多名玩家（多方）的票证。有效的选项包括最小值（min）玩家属性、最大值（max）玩家属性以及队伍中玩家所有玩家属性的平均值（avg）。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，并对队伍的技能水平进行平均值。

```
{
    "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该批次的第一张票据之间的距离对一批对战票证进行排序。

距离排序规则属性

- **sortDirection** – 对对战票证进行排序的方向。有效选项包括：
- **sortAttribute**– 用于对票证进行排序的玩家属性。
- **mapKey**– 如果是地图，则用于对玩家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张票证，请找到值最低的密钥。
  - `maxValue`– 对于添加到批次中的第一张票证，请找到值最高的密钥。
- **`partyAggregation`** – 该值决定 FlexMatch 如何处理多名玩家（多方）的票证。有效的选项包括票证玩家的最小值（`min`）、最大值（`max`）和平均（`avg`）值。默认为 `avg`。

FlexMatch 属性表达式

属性表达式在规则集中用于引用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。属性表达式通常采用以下两种形式之一：

- 单个玩家数据
- 计算出的个人玩家数据集合。

常见的对战属性表达式

有效属性表达式标识单个玩家、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	含义	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有效属性表达式：

Expression	含义	结果类型
`teams[red].players[playerId]`	红队所有玩家的玩家 ID	List<string>
`teams[red].players.attributes[skill]`	红队所有玩家的“技能”属性	List<number>
`teams[red,blue].players.attributes[skill]`	红队和蓝队所有玩家的“技能”属性，按队伍分组	List<List<number>>
`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	含义	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

聚合	输入	含义	输出
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：

Expression	含义	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(teams[*].players.attributes[skill]	对战游戏中的每个团队的平均技能	List<number>

Expression	含义	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 对战活动

Amazon GameLift FlexMatch 会在处理每张对战票证时发出事件。您可以将这些事件发布到 Amazon SNS 主题，如设置 FlexMatch 事件通知中所述。这些事件还会尽力近乎实时地发送到 Amazon CloudWatch Events Amazon CloudWatch Events。

本主题介绍了 FlexMatch 事件的结构，并提供了每种事件类型的示例。有关对战票证状态的更多信息，请参阅《Amazon GameLift API 参考》中的 MatchmakingTicket。

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"
  }
}
```

AWS 术语表

有关最新的 AWS 术语，请参阅 AWS 词汇表 参考中的 AWS 词汇表。