aws

FlexMatch デベロッパーガイド

# Amazon GameLift

Version

# Amazon GameLift: FlexMatch デベロッパーガイド

# Table of Contents

# Amazon とは GameLift FlexMatch

Amazon GameLift FlexMatch は、マルチプレイヤーゲーム用のカスタマイズ可能なマッチメーキングサービスです。を使用すると FlexMatch、ゲームのマルチプレイヤーマッチがどのように見えるかを定義するカスタムルールセットを構築し、試合ごとに互換性のあるプレイヤーを評価して選択する方法を決定できます。ゲームのニーズに合わせてマッチメーキングアルゴリズムの主要な側面を微調整することもできます。

をスタンドアロンのマッチメーキングサービス FlexMatch として使用するか、Amazon GameLift ゲームホスティングソリューションと統合します。例えば、 peer-to-peer アーキテクチャを持つゲームや、他のクラウドコンピューティングソリューションを使用するゲームで、スタンドアロン機能 FlexMatch として を実装できます。または、Amazon GameLift マネージド EC2 ホスティングまたは Amazon によるオンプレミスホスティング FlexMatch に を追加することもできます GameLift Anywhere。このガイドでは、特定のシナリオの FlexMatch マッチメーキングシステムを構築する方法について詳しく説明します。

FlexMatch では、ゲームの要件に応じてマッチメーキングの優先順位を柔軟に設定できます。例えば、次のオペレーションを実行できます。

- 試合のスピードとクオリティのバランスを見つけましょう。マッチルールを設定して、十分に良いマッチを素早く見つけたり、最適なプレイヤーエクスペリエンスのためにプレイヤーが最適なマッチを見つけるのに少し長く待たせます。
- よくマッチした選手やよくマッチしたチームに基づいて試合を行います。スキルや経験など、すべてのプレイヤーが類似の特性を持つ試合を作成します。または、各チームの特性が共通基準を満たしている一致を形成します。
- プレイヤーのレイテンシーがマッチメーキングにどのように影響するかに優先順位を付けます。すべてのプレイヤーのレイテンシーにハード制限を設定するか、試合の全員がレイテンシーが似ている限り、より高いレイテンシーを許容できるか。

> で作業を開始する準備はできました FlexMatchか？
>
> でゲームを起動して実行するための step-by-step ガイダンスについては FlexMatch、以下のトピックを参照してください。
>
> - FlexMatch と Amazon GameLift ホスティングの統合
> - スタンドアロンマッチメーキングのための Amazon GameLift FlexMatch 統合

# 主な FlexMatch 機能

スタンドアロンサービスとして を使用する場合 FlexMatchでも、Amazon GameLift ゲームホスティングで を使用する場合でも、すべての FlexMatch シナリオで次の機能を使用できます。

- [Customizable player matching.] ( カスタマイズ可能なプレイヤーマッチング。 ) プレイヤーに提供するすべてのゲームモードに合わせてマッチメーカーをデザインして構築しましょう。キープレイヤー属性 (スキルレベルやロールなど) と、ゲームにおいてよいプレイヤーマッチングを形成するための地理的なレイテンシー データを評価するためのカスタムルールを構築します。
- [Latency-based matching] ( レイテンシーに基づくマッチング ) プレイヤーのレイテンシー データを提供し、試合中のプレイヤーが同様の応答時間を持つことを要求する対戦ルールを作成します。この特徴は、プレイヤーのマッチメーキングプールが複数の地理的地域にまたがる場合に便利です。
- [Support for match sizes up to 200 players] ( 最大 200 人のプレイヤーの試合規模のSupport ) ゲーム用にカスタマイズされた対戦ルールを使用して、最大 40 人のプレイヤーの試合を作成します。合理化されたカスタムマッチングプロセスを使用して、プレイヤー待ち時間を管理しやすくするマッチングプロセスを使用して、最大200人のプレーヤーの試合を作成します。
- [Player acceptance] ( プレイヤーの承諾 ) 試合を確定してゲームセッションを開始する前に、提案された試合へのオプトインをプレイヤーに要求します。この機能を使用して、試合の新しいゲームセッションを配置 FlexMatch する前に、カスタム承諾ワークフローを開始し、プレイヤーの応答を に報告します。すべてのプレイヤーがマッチを受け入れるわけではない場合、提案されたマッチは失敗し、承諾したプレイヤーは自動的にマッチメーキングプールに戻ります。

  [Player parties support] ( プレイヤーパーティのサポート) 同じチームでプレイすることを希望するプレイヤーのグループに対してマッチングを生成します。 FlexMatch を使用して、必要に応じて試合を満たす追加のプレイヤーを検索します。
- [Expandable matching rules] ( 拡張可能なマッチングルール ) 成功した試合を見つけることなく一定の時間が経過すると、徐々に試合の要件を緩和します。プレイヤーがより迅速にプレイ可能なゲームに参加できるように、ルール拡張によって最初のマッチのルールを緩和する場所と時期を決めることができます。
- [Match backfill] ( バックフィルの一致 ) 既存のゲームセッションの空のプレイヤースロットを、最適な新しいプレイヤーで満たします。新しいプレイヤーをリクエストするタイミングと方法をカスタマイズし、同じカスタムマッチルールを使用して追加のプレイヤーを探します。

# FlexMatch Amazon GameLift ホスティングでの

FlexMatch には、Amazon でホストしているゲームで使用する以下の追加機能があります GameLift。これには、カスタムゲームサーバーまたはリアルタイムサーバーを使用したゲームが含まれます。

- [Game session placement] ( ゲームセッションの配置) 試合が正常に行われると、 は Amazon に新しいゲームセッション配置 FlexMatch を自動的にリクエストします GameLift。プレイヤー IDs やチーム割り当てなど、マッチメーキング中に生成されたデータはゲームサーバーに提供されるため、その情報を使用して試合のゲームセッションを開始できます。 FlexMatch その後、ゲームクライアントがゲームに参加できるように、ゲームセッション接続情報を渡します。試合中のプレイヤーが経験するレイテンシーを最小限に抑えるために、Amazon によるゲームセッション配置 GameLift では、提供されている場合は、リージョンレベルのプレイヤーレイテンシーデータを使用することもできます。
- [Automatic match backfill] ( 自動的な試合のバックフィル ) この機能を有効にすると、新しいゲームセッションが未入力のプレイヤースロットで開始されると、 は自動的にマッチバックフィルリクエスト FlexMatch を送信します。マッチメーキングシステムは最低限のプレイヤー数でゲームセッションプレイスメントプロセスを開始し、残りのスロットをすばやく埋めます。自動バックフィルを使って、マッチしたゲームセッションから脱落したプレイヤーを置き換えることはできません。

Amazon Elastic Compute Cloud (Amazon EC2) リソースでホストされているゲームで Amazon GameLift FleetIQ を使用する場合は、スタンドアロンサービス FlexMatch として を実装します。

# Amazon の料金 GameLift FlexMatch

Amazon では、インスタンスの使用期間別の料金と、転送されるデータ量別の帯域幅に対して GameLift 料金が発生します。Amazon GameLift サーバーでゲームをホストする場合、Amazon の料金に FlexMatch 使用量が含まれます GameLift。別のサーバーソリューションでゲームをホストする場合、 FlexMatch 使用料は別途請求されます。Amazon の料金と料金の完全なリストについては GameLift、「Amazon GameLift の料金」を参照してください。

Amazon でのゲームやマッチメーキングのコストの計算については GameLift、「 の使用方法を説明する GameLift」「Amazon 料金見積りの生成」を参照してくださいAWS Pricing Calculator。

# Amazon GameLift FlexMatch の仕組み

このトピックでは、FlexMatch システムのコアコンポーネントとそれらの相互作用方法を含む Amazon GameLift FlexMatch サービスの概要について説明します。

FlexMatch は、Amazon GameLift マネージドホスティングを使用するゲーム、または別のホスティングソリューションを使用するゲームで使用できます。リアルタイムサーバーを含む Amazon GameLift サーバーでホストされているゲームは、統合された Amazon GameLift サービスを使用して、利用可能なゲームサーバーを自動的に検索し、試合のゲームセッションをスタートします。FlexMatch をスタンドアロンサービスとして使用するゲーム ( Amazon GameLift FleetIQ を含む ) は、既存のホスティングシステムと調整して、ホスティングリソースを割り当てて、試合のゲームセッションをスタートする必要があります。

ゲームの FlexMatch の設定に関する詳細なガイダンスについては、FlexMatch の開始 を参照してください。

## マッチメーキングコンポーネント

FlexMatch マッチメーキングシステムには、次のコンポーネントの一部またはすべてが含まれます。

Amazon GameLift コンポーネント

これらは、FlexMatch サービスがゲームのマッチメーキングを実行する方法を制御する Amazon GameLift リソースです。これらは、コンソールや AWS CLI を含んだ Amazon GameLift ツールを使用するか、または、その代わりにプログラムによる Amazon GameLift 用の AWS SDK を使用して、作成および保守されます。

- FlexMatch マッチメーキング設定 (マッチメーカーとも呼ばれます)— マッチメーカーとは、ゲームのマッチメーキングプロセスをカスタマイズする設定値のセットです。ゲームには、複数のマッチメーカーがあり、それぞれが異なるゲームモードまたはエクスペリエンスに応じて構成されます。ゲームが FlexMatch にマッチメーキングリクエストを送信するときに、使用するマッチメーカーを指定します。
- [FlexMatch matchmaking rule set] ( FlexMatch マッチメーキングルールセット— ルールセットには、潜在的な試合のプレイヤーを評価し、承認または却下するために必要なすべての情報が含まれています。ルールセットは、試合のチーム構造を定義し、評価に使用されるプレーヤー属性を宣言し、受け入れ可能な試合の条件を記述するルールを提供します。ルールは、個別のプレイヤー、チーム、または試合全体に適用できます。たとえば、ルールによって、試合内のすべてのプレイヤーが同じゲームマップを選択することを要求したり、すべてのチームが同程度のプレイヤースキル平均を有していることを要求する場合があります。

- Amazon GameLift ゲームセッションキュー (Amazon GameLift マネージドホスティングでの FlexMatch の場合のみ) – ゲームセッションキューでは、利用可能なホストリソースを検索し、試合の新しいゲームセッションを開始します。キューの設定により、Amazon GameLift で利用可能なホストリソースを探す場所、および試合に利用できる最適なホストを選択する方法を決定します。

### カスタムコンポーネント

次のコンポーネントには、ゲームのアーキテクチャに基づいて実装する必要がある完全な FlexMatch システムに必要な機能が含まれています。

- [Player interface for matchmaking] ( マッチメーキング用のプレイヤーインターフェース ) — このインターフェースにより、プレイヤーは試合に参加できます。少なくとも、クライアントマッチメーキングサービスコンポーネントを通じてマッチメーキングリクエストを開始し、マッチメーキングプロセスに必要なスキルレベルやレイテンシー データなどのプレイヤー固有のデータを提供します。

  > **Note**
  >
  > ベストプラクティスとして、FlexMatch サービスとの通信は、ゲーム クライアントからではなく、バックエンドサービスによって行う必要があります。

- [Client matchmaking service] ( クライアントマッチメーキングサービス ) — このサービスは、プレイヤーインターフェイスからのプレイヤー参加リクエストをフィールド化し、マッチメーキングリクエストを生成し、FlexMatch サービスに送信します。処理中のリクエストについては、マッチメーキングイベントをモニタリングし、マッチメーキングステータスを追跡し、必要に応じてアクションを実行します。ゲームでのゲームセッションホスティングの管理方法によっては、このサービスはゲームセッションの接続情報をプレイヤーに返す場合があります。このコンポーネントでは FlexMatch サービスと通信するための Amazon GameLift API による AWS SDK を使います。
- [Match placement service (for FlexMatch as a standalone service only)] ( マッチプレースメントサービス ( スタンドアロンサービスとしての FlexMatch の場合のみ ) ) — このコンポーネントは、既存のゲームホスティングシステムと連携して、利用可能なホスティングリソースを見つけ、試合の新しいゲームセッションをスタートします。コンポーネントは、マッチメーキング結果を取得し、新しいゲームセッションのスタートに必要な情報 ( 試合内のすべてのプレイヤーのプレイヤー ID、属性、チーム割り当てなど ) を抽出する必要があります。

# FlexMatch マッチメーキングプロセス

このトピックでは、ベーシックなマッチメーキングシナリオと、さまざまなゲームコンポーネントと FlexMatch サービス間の相互作用について説明します。

プレイヤーのマッチメーキングのリクエスト

ゲームクライアントを使用しているプレイヤーが「ゲームに参加」ボタンをクリックします。このアクションにより、クライアントのマッチメーキングサービスが FlexMatch にマッチメーキングリクエストを送信します。リクエストは、リクエストを実行する際に使用する FlexMatch マッチメーカーを識別します。リクエストには、スキルレベル、プレイ設定、地理的なレイテンシーデータなど、カスタムマッチメーカーが必要とするプレイヤー情報も含まれます。1人のプレイヤーまたは複数のプレイヤーに対してマッチメーキングリクエストを行うことができます。

マッチメーキング プールにリクエストを追加

FlexMatch がマッチメーキングリクエストを受信すると、マッチメーキングチケットを生成し、マッチメーカーのチケットプールに追加します。チケットは、試合が始まるか、それともマッチメーカーの制限時間に達するかまでプールに残ります。クライアントのマッチメーキングサービスには、チケットステータスの変更など、マッチメーキングイベントについて定期的に通知されます。

試合を構築する

FlexMatch マッチメーカーは、プール内のすべてのチケットに対して次のプロセスを継続的に実行します。

1. マッチメーカーはチケット年齢でプールをソートし、最も古いチケットから潜在的な試合の構築を開始します。
2. マッチメーカーは潜在的な試合に 2 番目のチケットを追加し、カスタムマッチメーキングルールに対して結果を評価します。潜在的な試合が評価に合格すると、チケットのプレイヤーはチームに割り当てられます。
3. マッチメーカーは次のチケットを順番に追加し、評価プロセスを繰り返します。すべてのプレイヤーのスロットがいっぱいになると、試合は準備完了です。

ラージな試合 ( 41〜200人 ) のマッチメーキングでは、合理的な時間枠で試合を構築できるように、上記プロセスの修正バージョンを使用します。マッチメーカーは、各チケットを個別に評価する代わりに、事前にソートされたチケットプールを潜在的な試合に分割し、指定したプレイヤーの特性に基づいて各試合のバランスをとります。たとえば、マッチメーカーが類似する低レイテンシーの場所に基づいてチケットを事前にソートし、試合後のバランシングを使用して、チームがプレイヤーのスキルで均等に試合するようにします。

### マッチメーキングの結果の報告

受け入れ可能な試合が見つかると、一致したすべてのチケットが更新され、一致したチケットごとに成功したマッチメーキングイベントが生成されます。

- スタンドアロンサービスとしての FlexMatch: ゲームはマッチメーキングイベントで試合結果を受け取ります。結果データには、マッチングしたすべてのプレイヤーとそのチームの割り当てのリストが含まれます。試合リクエストにプレイヤーのレイテンシー情報が含まれている場合、結果には試合に最適な地理的位置が示されます。
- Amazon GameLift ホスティングソリューションを使用した FlexMatch: 試合の結果は、ゲームセッションプレイスメントのために Amazon GameLift キューに自動的に渡されます。マッチメーカーは、ゲームセッションの配置に使用されるキューを決定します。

### 試合のゲームセッションのスタート

提案された試合が正常に形成されると、新しいゲームセッションが開始されます。ゲームサーバーは、試合のゲームセッションを設定する際に、プレイヤー ID やチーム割り当てなどのマッチメーキング結果データを使用できます。

- スタンドアロンサービスとしてのFlexMatch：カスタムマッチプレースメントサービスは、正常なマッチメーキングイベントから試合結果データを取得し、試合に使用可能なホスティング リソースを割り当てるために既存のゲーム セッション プレースメントシステムに接続します。ホスティング リソースが見つかると、マッチプレースメントサービスは既存のホスティングシステムと調整して、新しいゲームセッションをスタートし、接続情報を取得します。
- Amazon GameLiftホスティング ソリューションを使用した FlexMatch: ゲームセッション キューは、マッチに最適なゲームサーバーを特定します。キューの構成方法に応じて、ゲームセッションを最低コストのリソースで配置し、プレイヤーのレイテンシーが低い場所（プレイヤーのレイテンシー データが提供されている場合）を配置しようとします。ゲームセッションが正常に配置されると、Amazon GameLift サービスは、マッチメーキング結果やその他のオプションのゲームデータを渡して、新しいゲームセッションを開始するようにゲームサーバーに要求します。

### プレイヤーを試合にConnectする

ゲームセッションが開始されると、プレイヤーはセッションにConnectし、チームの割り当てをクレームし、ゲームプレイを開始します。

- スタンドアロンサービスとしての FlexMatch: ゲームは既存のゲームセッション管理システムを使用して、接続情報をプレイヤーに返します。

- Amazon GameLift ホスティングソリューションを使用した FlexMatch: ゲームセッションプレイスメントが正常に行われると、FlexMatch はゲームセッション接続情報とプレイヤーセッション ID を使用して、一致したチケットをすべて更新します。

# FlexMatch サポートされる AWS リージョン

Amazon GameLift ホスティングソリューション FlexMatch で を使用している場合は、ゲームをホストしている任意の場所でマッチングされたゲームセッションをホストできます。Amazon GameLift ホスティング の AWS リージョンと ロケーションの完全なリストを参照してください。

すべての FlexMatch ユーザーについて、次のサポートされている で、マッチメーキング設定やルールセットなどの FlexMatch リソースをホストできますAWS リージョン。マッチメーカーの ロケーションを選択する を参照してください。

| AWS リージョン 名 | リージョンコード |
|---|---|
| 米国東部 ( バージニア北部 ) | us-east-1 |
| 米国西部 ( オレゴン ) | us-west-2 |
| アジアパシフィック ( ソウル ) | ap-northest-2 |
| アジアパシフィック ( シドニー ) | ap-southeast-2 |
| アジアパシフィック (東京) | ap-northeast-1 |
| 欧州 (フランクフルト) | eu-central-1 |
| 欧州 (アイルランド) | eu-west-1 |
| 中国 (北京および寧夏リージョン ) | |

# FlexMatch の設定

Amazon GameLift FlexMatch は AWS サービスで、このサービスを利用するための AWS アカウントが必要です。AWS アカウントの作成は無料です。AWS アカウントで何ができるかの詳細については、「AWS の使用開始」を参照してください。

他の Amazon GameLift ソリューションと FlexMatch を使用している場合は、以下のトピックを参照してください。

- Amazon GameLift ホスティングとリアルタイムサーバーへのアクセスを設定
- Amazon GameLift FleetIQ を使用して Amazon EC2 でホスティングするためのアクセスをセットアップ

Amazon GameLift アカウントのセットアップするには

1. [Get an account]. ( アカウントを取得します ) 。Amazon Web Services を開き、コンソールにサインインする を選択します。プロンプトに従って新しいアカウントを作成するか、既存のアカウントにサインインします。
2. 管理者ユーザーグループをセットアップします。AWS Identity and Access Management (IAM) サービスコンソールを開き、手順に従ってユーザーまたはユーザーグループを作成または更新します。IAM は、AWS サービスとリソースへのアクセスを管理します。Amazon GameLift コンソールを使用して、または Amazon GameLift API を呼び出して FlexMatch リソースにアクセスするすべての人には、明示的なアクセス権が付与されている必要があります。コンソール (または AWS CLI やその他のツール) を使用してユーザーグループをセットアップする方法の詳細については、「IAM ユーザーの作成」を参照してください。
3. アカウントのユーザーまたはグループにアクセス権限ポリシーを付与する。AWSサービスとリソースへのアクセスは、IAM ポリシー をユーザーまたはユーザーグループに付与することで管理されます。権限ポリシーは、ユーザーがアクセス権を持つ必要がある AWS サービスとアクションのセットを指定します。

   Amazon GameLift の場合は、カスタムアクセス許可ポリシーを作成し、各ユーザーまたはユーザーグループに付与する必要があります。ポリシーは JSON ドキュメントです。以下の例を使用して、ポリシーを作成します。

次の例は、すべての Amazon GameLift リソースとアクションの管理アクセス許可を持つインラインアクセス許可ポリシーを示しています。FlexMatch 固有の項目だけを指定して、アクセスを制限するように選択できます。

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

# FlexMatch の開始

このセクションのリソースの使用は、FlexMatch でマッチメーキングシステムの構築をスタートするのに役立ちます。

トピック

- スタンドアロンマッチメーキングのための Amazon GameLift FlexMatch 統合
- FlexMatch と Amazon GameLift ホスティングの統合

## スタンドアロンマッチメーキングのための Amazon GameLift FlexMatch 統合

このトピックでは、FlexMatch をスタンドアロンのマッチメーキングサービスとして実行するための完全な統合プロセスの概要を説明します。このプロセスは、マルチプレイヤーゲームがピアツーピア、カスタム設定されたオンプレミスハードウェア、またはその他のクラウドコンピューティングプリミティブを使用してホストされている場合に使用します。このプロセスは、Amazon EC2 でホストされているゲームのホスティング最適化ソリューションである Amazon GameLift FleetIQ でも使用できます。Amazon GameLift マネージドホスティング (リアルタイムサーバーを含む) を使用してゲームをホストする場合は、FlexMatch と Amazon GameLift ホスティングの統合 を参照してください。

統合を開始する前に、AWS アカウントを作成し、Amazon GameLift サービスのアクセス権限を設定する必要があります。詳細については、「FlexMatch の設定」を参照してください。Amazon GameLift FlexMatch マッチメーカーおよびルールセットの作成と管理に関連する重要なタスクはすべて、Amazon GameLift コンソールを使用して実行できます。

1. FlexMatch マッチメーキングルールセットの作成 カスタムルールセットには、試合の構築方法に関する完全な手順が記載されています。ここでは、各チームの構造とサイズを定義します。また、試合が有効になるために満たす必要がある一連の要件も提供します。FlexMatch は、これを試合にプレーヤーを含めるか除外するために使用します。これらの要件は、個々のプレイヤーに適用される場合があります。ルールセットで FlexMatch アルゴリズムをカスタマイズすることもできます。たとえば、最大 200 人のプレイヤーによる大規模対戦を構築できます。以下のトピックを参照してください。

   - FlexMatch ルールセットの構築
   - FlexMatch ルールセットの例

2. イベントの通知を設定します。通知を使用して、保留中の対戦リクエストのステータスを含む FlexMatch マッチメーキングアクティビティを追跡します。これは、提案された試合の結果を提供するために使用されるメカニズムです。マッチメーキングリクエストは非同期であるため、リクエストのステータスを追跡する方法が必要です。その手段としては、通知が最適です。以下のトピックを参照してください。

   - FlexMatch イベント通知をセットアップする
   - FlexMatch マッチメーキングイベント

3. FlexMatch マッチメーキング設定を設定します。マッチメーカーとも呼ばれ、このコンポーネントはマッチメーキングリクエストを受信して処理します。マッチメーカーを設定するには、ルールセット、通知ターゲット、および最大待機時間を指定します。オプション機能を有効にすることもできます。以下のトピックを参照してください。

   - FlexMatch マッチメーカーの設計
   - マッチメーキング設定の作成

4. クライアントマッチメーキングサービスを構築します。FlexMatch にマッチメーキングリクエストを構築して送信する機能を備えたゲームクライアントサービスを作成または拡張します。マッチメーキングリクエストを構築するには、このコンポーネントに、マッチメーキングルールセットに必要なプレイヤーデータと、オプションでリージョンのレイテンシー情報を取得するメカニズムが必要です。また、リクエストごとにユニークなチケット ID を作成して割り当てるメソッドが必要です。また、プレイヤーが提案された試合へのオプトインを要求するプレイヤー受け入れワークフローを構築することもできます。また、このサービスは、マッチメーキングイベントを監視して、マッチ結果を取得し、成功したマッチのゲームセッション配置を開始する必要があります。次のトピックを参照してください。

   - ゲーム クライアントへの FlexMatch の追加

5. マッチプレースメントサービスを構築します。既存のゲームホスティングシステムと連動するメカニズムを作成して、利用可能なホスティングリソースを見つけ、試合を成功させるために新しいゲームセッションを開始します。このコンポーネントは、対戦結果情報を使用して、使用可能なゲームサーバーを取得し、試合の新しいゲームセッションを開始できることが必要です。また、マッチバックフィルリクエストを行うワークフローを実行することもできます。マッチバックフィルリクエストでは、マッチメーキングを使用して、すでに実行中のマッチしたゲームセッション内の空きスロットを埋めることができます。

# FlexMatch と Amazon GameLift ホスティングの統合

FlexMatch は、カスタムゲームサーバーおよびリアルタイムサーバーのマネージド Amazon GameLift ホスティングで使用できます。FlexMatch マッチメーキングをゲームに追加するには、以下のタスクを行う必要があります。

- マッチメーカーを設定します。マッチメーカーは、プレイヤーからマッチメーキングリクエストを受信して処理します。定義されたルールのセットに基づいてプレイヤーをグループ化し、マッチングが成功するごとに、新しいゲームセッションとプレイヤーセッションを作成します。マッチメーカーをセットアップするには、以下の手順を実行します。
  - ルールセットを作成します。ルールセットはマッチメーカーに有効なマッチを作成する方法を指定します。チーム構造を指定し、マッチングに含めるプレイヤーの評価方法を指定します。以下のトピックを参照してください。
    - FlexMatch ルールセットの構築
    - FlexMatch ルールセットの例
  - ゲームセッションキューを作成します。キューは、各マッチングの最適なリージョンを見つけ、そのリージョンで新しいゲームセッションを作成します。既存のキューを使用するか、マッチメーキング用に新しいキューを作成します。次のトピックを参照してください。
    - キューを作成する
  - 通知を設定します (オプション)。マッチメーキングリクエストは非同期であるため、リクエストのステータスを追跡する方法が必要です。その手段としては、通知が最適です。次のトピックを参照してください。
    - FlexMatch イベント通知をセットアップする
  - マッチメーカーを設定します。ルールセット、キュー、および通知ターゲットの準備ができたら、マッチメーカーの設定を作成します。以下のトピックを参照してください。
    - FlexMatch マッチメーカーの設計
    - マッチメーキング設定の作成
- FlexMatch をゲームクライアントサービスに統合します。マッチメーキングを使用して新しいゲームセッションを開始するために、ゲームクライアントサービスに機能を追加します。マッチメーキングのリクエストでは、使用するマッチメーカーを指定し、マッチングに必要なプレイヤーデータを提供します。次のトピックを参照してください。
  - ゲーム クライアントへの FlexMatch の追加
- FlexMatch をゲームサーバーに統合します。マッチメーキングを通して作成されたゲームセッションを開始するために、ゲームサーバーに機能を追加します。このタイプのゲームセッションのリ

クエストには、プレイヤーとチームの割り当てを含むマッチング固有の情報が含まれます。ゲームサーバーは、ゲームセッションをマッチングのために構築する際に、この情報にアクセスして使用する必要があります。次のトピックを参照してください。

- Amazon GameLift ホストゲームサーバーへの FlexMatch の追加

- FlexMatch バックフィルを設定します (オプション)。既存のゲームの空きプレイヤースロットを埋める追加のプレイヤーマッチングをリクエストします。自動バックフィルを有効にして、Amazon GameLift によるバックフィルリクエストの管理ができます。または、ゲームクライアントまたはゲームサーバーに機能を追加してバックフィルを手動で管理することにより、バックフィルを手動で管理できます。次のトピックを参照してください。
  - FlexMatch を使用して既存のゲームをバックフィル

> **Note**
>
> 現在、FlexMatch バックフィルは、Realtime サーバーを使用しているゲームで使用することはできません。

# Amazon GameLift FlexMatch マッチメーカーの構築

FlexMatch マッチメーカープロセスでは、ゲームマッチングを構築します。受け取ったマッチメーキングリクエストのプールの管理、マッチングのチームの編成、最適なプレイヤーグループを見つけるためのプレイヤーの処理および選択、マッチングのゲームセッションを配置および開始するプロセスの起動を行います。このトピックでは、マッチメーカーの主な特徴と、ゲームに合わせて設定をカスタマイズする方法について説明します。

FlexMatch マッチメイカーがマッチメーキングリクエストを処理する方法の詳細については、「FlexMatch マッチメーキングプロセス」を参照してください。

トピック



# FlexMatch マッチメーカーの設計

このトピックでは、ゲームに合ったマッチメーカーを設計する方法についてのガイダンスを提供します。

## ベーシックなマッチメーカーの設定

マッチメーカーには次の要素が最低限必要です：

- [rule set](ルールセット)は、マッチングのチームのサイズと範囲を決定し、マッチングのプレイヤーの評価に使用するルールのセットを定義します。各マッチメーカーは 1 つのルールセットを使用するように設定されます。「FlexMatch ルールセットの構築」および「FlexMatch ルールセットの例」を参照してください。
- [notification target] ( 通知ターゲット はすべてのマッチメーキングイベント通知を受信します。Amazon Simple Notification Service (SNS) トピックを設定し、マッチメーカにトピック ID を追加する必要があります。通知の設定の詳細については、「FlexMatch イベント通知をセットアップする」を参照してください。

- [request timeout](リクエストのタイムアウト)は、マッチメーキングリクエストがリクエストプールに残留できる期間を決定します。この期間内にリクエストはマッチングの候補として評価されます。リクエストがタイムアウトすると、マッチングの対象外となり、プールから削除されます。
- Amazon GameLift マネージドホスティング FlexMatch で を使用する場合、ゲームセッションキューは、試合のゲームセッションをホストするための最適なリソースを見つけ、新しいゲームセッションを開始します。各キューには、ゲームセッションを配置できる場所を決定するロケーションとリソースタイプ (スポットインスタンスまたはオンデマンドインスタンスを含む) のリストが設定されます。キューの詳細については、「マルチリージョンキューの使用」参照してください。。

## マッチメーカーのロケーションを選択する

マッチメーキングアクティビティを実行する場所を決定し、その場所にマッチメーキング設定とルールセットを作成します。Amazon は、ゲームの試合リクエストのチケットプール GameLift を保持し、有効な試合をソートして評価します。試合を行うと、Amazon はゲームセッション配置の試合の詳細 GameLift を送信します。マッチングされたゲームセッションは、ホスティングソリューションでサポートされている任意の場所で実行できます。

FlexMatch リソースを作成できる場所FlexMatch サポートされる AWS リージョンについては、「」を参照してください。

マッチメーカーAWS リージョンの を選択するときは、ロケーションがパフォーマンスにどのように影響するか、プレイヤーのマッチエクスペリエンスを最適化する方法を考慮してください。推奨されるベストプラクティスを以下に示します：

- マッチメーカーは、マッチ FlexMatch メーキングリクエストを送信するプレイヤーやクライアントサービスに近いロケーションに配置します。この方法により、マッチメーキングリクエストワークフローに対するレイテンシーの影響が減少し、効率が向上します。
- ゲームがグローバルオーディエンスに到達する場合は、複数のロケーションでマッチメーカーを作成し、プレイヤーに最も近いマッチメーカーで試合のリクエストをルーティングすることを検討してください。これにより、効率を高めるだけでなく、地理的に近接しているプレイヤーとチケットプールが形成され、レイテンシー要件に基づいてマッチメイカーがプレイヤーをマッチングする能力が向上します。
- Amazon GameLift マネージドホスティング FlexMatch で を使用する場合は、マッチメーカーと使用するゲームセッションキューを同じ場所に配置します。これにより、マッチメーカーとキュー間の通信レイテンシーを最小限に抑えることができます。

# オプション要素の追加

これらの最小要件に加えて、以下の追加のオプションを使用してマッチメーカーを設定できます。Amazon GameLift ホスティングソリューション FlexMatch で を使用している場合は、多くの機能が組み込まれています。をスタンドアロンのマッチメーキングサービス FlexMatch として使用している場合は、これらの機能をシステムに組み込むことができます。

プレイヤーの承諾

マッチング候補として選択されたすべてのプレイヤーに参加の承諾を要求するようにマッチメーカーを設定できます。承諾を要求する場合は、提案した試合を承諾または却下するオプションをすべてのプレイヤーに提供する必要があります。マッチングを完了するには、マッチング案のすべてのプレイヤーから事前に承諾を受け取る必要があります。いずれかのプレイヤーが試合を却下するか、承諾に失敗すると、提案した試合は破棄され、チケットは次のように処理されます。すべてのプレイヤーが試合を承諾したチケットは、マッチメーキング プールに返され、処理が続行されます。少なくとも 1 人のプレイヤーが試合を拒否するか、応答しなかったチケットは違反ステータスになり、処理が中断されます。プレイヤーの承諾には制限時間が必要です。試合の続行にはすべてのプレイヤーが制限時間内に提案した試合を承諾することが必要です。

バックフィルモード

FlexMatch バックフィルを使用して、ゲームセッションの存続期間中、ゲームセッションが十分に一致した新しいプレイヤーで満たされます。バックフィルリクエストを処理する場合、 は元のプレイヤーのマッチングに使用されたのと同じマッチメーカー FlexMatch を使用します。バックフィルチケットを新しい試合のチケットで優先させる方法をカスタマイズして、バックフィルチケットをラインの先頭または末尾のいずれかに配置できます。つまり、新しいプレイヤーがマッチメーキング プールを入力すると、新規に形成されたゲームではなく、既存のゲームに配置される可能性が高くなります。

手動バックフィルは、ゲームがマネージド Amazon GameLift ホスティングまたは他のホスティングソリューション FlexMatch で を使用するかどうかにかかわらず使用できます。手動バックフィルでは、バックフィルリクエストをいつトリガーするかを柔軟に決定できます。たとえば、ゲームの特定のフェーズ中や、特定の条件が存在するときのみ、新しいプレイヤーを追加したい場合があるかもしれません。

自動バックフィルは、マネージド Amazon GameLift ホスティングを使用するゲームでのみ使用できます。この機能を有効にすると、ゲームセッションが開いているプレイヤースロットで始まる場合、Amazon はバックフィルリクエストの自動生成 GameLift を開始します。この特徴を使用すると、新しいゲームが最小限のプレイヤー数で開始され、新しいプレイヤーがマッチメーキング プー

ルに入力するとすぐに補充されるようにマッチメーキングを設定することができます。ゲームセッションの有効期間中は、いつでも自動バックフィルをオフにすることができます。

ゲームのプロパティ

Amazon GameLift マネージドホスティング FlexMatch で を使用するゲームの場合、新しいゲームセッションがリクエストされるたびにゲームサーバーに渡される追加情報を提供できます。これは、作成する試合のタイプに対してゲームセッションをスタートするために必要なゲームモードの設定を渡すのに便利な方法です。マッチメーカーによって作成された試合のすべてのゲームセッションでは、同じゲームプロパティセットを受け取ります。異なるマッチメーキング構成を作成することで、ゲームのプロパティ情報を変えることができます。

プレイヤースロットの予約

各マッチングの特定のプレイヤースロットを予約し、将来の使用のために確保できます。これを行うには、マッチメーキング設定の "additional player count" プロパティを設定します。

カスタムイベントデータ

このプロパティを使用して、マッチメーカーのすべてのマッチメーキング関連イベントに一連のカスタム情報を含めます。この機能は、マッチメーカーのパフォーマンスを追跡するなど、ゲーム固有の特定のアクティビティを追跡するのに役立ちます。

# FlexMatch ルールセットの構築

FlexMatch マッチメーカーごとにルールセットが必要です。ルールセットは、マッチングの 2 つの重要な要素として、ゲームのチーム構造とサイズ、および最善のマッチングを実現するためにプレイヤーをグループ化する方法を決定します。

たとえば、ルールセットでは「5 名のプレイヤーで構成されるチームを 2 つ編成し、1 つのチームは防御者、別のチームは攻撃者として両チームのマッチングを作成する」というように定義できます。チームには初心者と経験豊富なプレイヤーが含まれる可能性がありますが、2 つのチームのスキル平均は 10 ポイント以内である必要があります。30 秒後にマッチングが作成されない場合は、スキルの要件を徐々に緩和します。

このセクションのトピックでは、マッチメイキングルールセットを設計、構築する方法について説明します。ルールセットを作成するときは、Amazon GameLift コンソールまたは AWS CLI を使用できます。

トピック

- FlexMatch ルールセットの設計
- ラージ対戦ルールセットの設計
- マッチメーキングルールセットの作成
- FlexMatch ルールセットの例
- FlexMatch ルール言語

## FlexMatch ルールセットの設計

このトピックでは、ルールセットの基本構造と最大 40 人のプレイヤーのマッチに使用するルールセットの構築方法について説明します。マッチメーキングルールセットは、2 つの操作を行います。1 つ目はマッチングのチーム構造とサイズを設計すること、2 つ目は可能な限り最良のマッチを形成するプレイヤーの選択方法をマッチメーカーに伝えることです。

マッチメーキングルールセットは他にもできることが多くあります。例えば、以下のことが可能です。

- ゲームのマッチメーキングアルゴリズムを最適化します。
- ゲームプレイの品質を保護するために、最小プレイヤーレイテンシー要件をセットアップします。
- 時間をかけてチームの要件とマッチルールを徐々に緩和していき、アクティブなプレイヤー全員が希望するマッチを見つけられるようにします。
- パーティーの集約を使用してグループのマッチメーキングリクエストの処理を定義します。
- 40 人以上のプレイヤーが集まる大規模なマッチを処理します。大きなマッチの構築に関する詳細については、「ラージ対戦ルールセットの設計」を参照してください。

マッチメーキングのルールセットを作成する際は、以下のオプションタスクと必須タスクを検討してください。

- ルールセットを記述する (必須)
- マッチアルゴリズムのカスタマイズ
- プレイヤー属性の宣言
- 対戦チームの定義
- プレイヤーマッチングのルール設定
- 時間の経過による要件の許可

ルールセットは、Amazon GameLift コンソールまたは `CreateMatchmakingRuleSet` オペレーションを使用して作成できます。

## ルールセットを記述する (必須)

ルールセットの詳細を指定します。

- name (オプション) – 自分で使用するためのわかりやすいラベル。この値は、Amazon GameLift でルールセットを作成するときに指定するルールセット名とは関連付けられていません。
- ruleLanguageVersion – FlexMatch ルールの作成に使用されるプロパティ式言語のバージョン。値は `1.0` にする必要があります。

## マッチアルゴリズムのカスタマイズ

FlexMatch は、ほとんどのゲームに対してデフォルトのアルゴリズムを最適化し、プレイヤーを最小限の待機時間で許容可能なマッチに入れます。ゲームのアルゴリズムをカスタマイズし、マッチメーキングを調整できます。

以下はデフォルトの FlexMatch マッチメーキングアルゴリズムです。

1. FlexMatch は、オープンマッチメーキングチケットとバックフィルチケットはすべてチケットプールに配置します。
2. FlexMatch は、プール内のチケットを 1 つ以上のバッチにランダムにグループ化します。チケットプールが大きくなると、FlexMatch は最適なバッチサイズを維持するために追加のバッチを作成します。
3. FlexMatch は、各バッチ内のチケットを経過時間別にソートします。
4. FlexMatch は、各バッチの最も古いチケットに基づいてマッチを作成します。

対戦アルゴリズムをカスタマイズするには、ルールセットスキーマに `algorithm` コンポーネントを追加します。詳細については、FlexMatch ルールセットスキーマ コマンドのリファレンスを参照してください。

次のオプションのカスタマイズは、マッチメーキングプロセスのさまざまな段階に影響します。

- 事前バッチソートの追加
- batchDistance 属性に基づいてバッチを形成する
- バックフィルチケットの優先順位付け

- 拡張時に古いチケットを優先

事前バッチソートの追加

バッチを作成する前にチケットプールをソートします。このタイプのカスタマイズは、大規模なチケットプールを持つゲームで最も効果的です。事前バッチソートは、マッチメーキングプロセスをスピードアップし、定義された特性においてプレイヤーの均一性を高めるのに役立ちます。

バッチ前のソート方法は、アルゴリズムプロパティbatchingPreference で定義します。デフォルトの設定は、random です。

事前バッチソートのカスタマイズには、次のオプションが含まれます。

- [Sort by player attributes.] ( プレイヤーの属性でソート。 ) チケットプールを事前ソートするプレイヤー属性のリストを提供します。

  プレイヤー属性でソートするには、batchingPreference を sorted に設定し、sortByAttributesでプレイヤー属性のリストを定義します。属性を使用するには、最初にルールセットの playerAttributes コンポーネント内で属性を宣言します。

  次の例では、FlexMatch はプレイヤーの優先ゲームマップに基づいてチケットプールをソートし、次にプレイヤーのスキル別にソートします。結果のバッチには、同じマップを使用したい類似のスキルのプレイヤーが含まれる可能性が高くなります。

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

- [Sort by latency](レイテンシーでソート) 可能な限りのレイテンシーでマッチを作成するか、許容できるレイテンシーでマッチを迅速に作成します。このカスタマイズは、40 人以上のプレイヤーが参加する大規模なマッチを形成するルールセットに役立ちます。

  アルゴリズムプロパティ strategy を balanced に設定します。バランス戦略では、使用できるルールステートメントのタイプが制限されます。詳細については、「ラージ対戦ルールセットの設計」を参照してください。

  FlexMatch は、次のいずれかの方法で、プレイヤーから報告されたレイテンシーデータに基づいてチケットをソートします。

- レイテンシーが最も低い場所。チケットプールは、プレイヤーが最低レイテンシー値を報告するロケーションによって事前にソートされています。その後、FlexMatch は同じ場所でチケットを低レイテンシーでバッチ処理するため、ゲームプレイのエクスペリエンスが向上します。また、各バッチのチケット数を減らすため、マッチメーキングに時間がかかることがあります。このカスタマイズを使用するには、次の例のように batchingPreference を fastestRegion に設定します。

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

- 許容可能なレイテンシーをすぐにマッチングする。チケットプールは、プレーヤーが許容可能なレイテンシー値を報告するロケーション別に事前にソートされています。これにより、より多くのチケットを含むバッチの数が減ります。各バッチでより多くのチケットがあると、許容可能なマッチを見つけるのが迅速になります。このカスタマイズを使用するには、次の例のようにプロパティ batchingPreference を largestPopulation に設定します。

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

> **Note**
>
> バランス戦略のデフォルト値は、largestPopulation です。

バックフィルチケットの優先順位付け

ゲームが自動バックフィルまたは手動バックフィルを実行している場合、FlexMatch はリクエストタイプに基づいてマッチメーキングチケットを処理する方法をカスタマイズできます。リクエストタイプは、新しいマッチリクエストでもバックフィルリクエストでもかまいません。デフォルトでは、FlexMatch はどちらのタイプのリクエストも同様に扱います。

バックフィルの優先順位付けは、FlexMatch がチケットをバッチ処理した後の処理方法に影響します。バックフィルの優先順位付けは、網羅的な検索戦略を使用するルールセットを必要とします。

FlexMatch は複数のバックフィルチケットをまとめてマッチングしません。

バックフィルチケットの優先順位を変更するには、プロパティ backfillPriority を設定します。

- バックフィルチケットを最初にマッチします。このオプションは、新しいマッチを作成する前に、バックフィルチケットのマッチングを試みます。つまり、参加するプレイヤーは、既存のゲームにスロットされる可能性が高くなります。

  ゲームで自動バックフィルを使用している場合は、これを使用するのが最適です。自動バックフィルは、ゲームセッション時間が短く、プレーヤーのターンアラウンドが高いゲームでよく使用されます。自動バックフィルは、FlexMatch がオープンスロットを埋めるためにより多くのプレイヤーを検索しても、これらのゲームが最小限の実行可能な試合を形成して開始するのに役立ちます。

  backfillPriority を high に設定します。

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

- バックフィルチケットを最後にマッチングする。このオプションでは、他のすべてのチケットが評価されるまで、バックフィルチケットは無視されます。つまり、FlexMatch は参加するプレイヤーを新しいゲームにマッチングできない場合にのみ、既存のゲームにバックフィルします。

  このオプションは、バックフィルを、新しい試合を形成できるプレーヤーが十分ではない場合などに、プレイヤーがゲームに参加できる最後のチャンスのオプションとして使用する場合に便利です。

  backfillPriority を low に設定します。

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

拡張時に古いチケットを優先

拡張ルールはマッチの完了が困難な場合に基準を緩和します。Amazon GameLift は、部分的に完了した試合のチケットが特定の期間に達したときに拡張ルールを適用します。チケットの作成タイム

スタンプによって、Amazon GameLift がルールを適用するタイミングが決まります。デフォルトでは、FlexMatch は最後に対戦したチケットのタイムスタンプを追跡します。

FlexMatch が拡張ルールを適用するタイミングを変更するには、プロパティ `expansionAgeSelection` を以下のように設定します。

- 最新のチケットに基づいて拡張します。このオプションは、潜在的な対戦に追加された最新のチケットに基づいて拡張ルールを適用します。FlexMatch が新しいチケットをマッチングするたびに、タイムクロックがリセットされます。このオプションを使うと、マッチの質が高くなる傾向がありますが、マッチングに時間がかかる傾向があります。マッチングに時間がかかりすぎると、完了する前にマッチリクエストがタイムアウトする場合があります。`expansionAgeSelection` を `newest` に設定します。`newest` はデフォルトです。
- 最も古いチケットに基づいて拡張する。このオプションは、マッチ候補の最も古いチケットに基づいて拡張ルールを適用します。このオプションを使用すると、FlexMatch は拡張をより速く適用するため、最も早くマッチングしたプレイヤーの待ち時間が改善されるますが、すべてのプレイヤーのマッチ品質が低下します。`expansionAgeSelection` を `oldest` に設定します。

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

## プレイヤー属性の宣言

このセクションでは、マッチメーキングリクエストに含める個々のプレイヤーの属性をリストアップします。ルールセットでプレイヤー属性を宣言する理由は 2 つあります。

- ルールセットにプレイヤー属性に依存するルールが含まれている場合。
- マッチリクエストを通じてプレイヤー属性をゲームセッションに渡したい場合。例えば、各プレイヤーが接続する前に、プレイヤーキャラクターの選択肢をゲームセッションに渡したい場合があります。

プレイヤー属性を宣言する際に、以下の情報を含めます。

- 名前 (必須) この値はルールセットごとにユニークであることが必要です。
- type (必須) 属性値のデータ型です。有効なデータ型は、数値、文字列、または文字列マップです。

- default (オプション) マッチメーキングリクエストが属性値を提供しない場合、使用するデフォルト値を入力します。デフォルトが宣言されておらず、リクエストに値が含まれていない場合、FlexMatch はリクエストを処理できません。

## 対戦チームの定義

マッチング用のチームの構造とサイズを記述します。各マッチングには少なくとも 1 つのチームが必要であり、チームの数は自由に定義できます。チームには同じ数のプレイヤーを含めることも、非対称とすることもできます。たとえば、プレイヤー 1 人のモンスターチームと、プレイヤー 10 人のハンターチームを定義できます。

FlexMatch はルールセットでのチームサイズの定義に基づいて、小規模な試合、またはラージな試合として対戦リクエストを処理します。最大 40 人のプレイヤーのマッチング案は小規模なマッチングで、40 人を超えるプレイヤーのマッチング案は大規模なマッチングです。ルールセットのマッチングサイズ案を定義するには、ルールセットに定義されたすべてのチームに対して maxPlayer 設定を追加します。

- 名前 (必須) 各チームにユニークな名前を割り当てます。この名前はルールや拡張で使用し、FlexMatch はゲームセッションのマッチメーキングデータを参照します。
- maxPlayers (必須) チームに割り当てるプレイヤーの最大数を指定します。
- minPlayers (必須) チームに割り当てるプレイヤーの最小数を指定します。
- quantity (オプション) — この定義で作成するチームの数を指定します。FlexMatch がマッチを作成すると、これらのチームには指定された名前が与えられ、その後に数字が付加されます。例: `Red-Team1`、`Red-Team2`、`Red-Team3`。

FlexMatch は、チームを最大プレイヤー数まで満たすよう試みますが、より少ないプレイヤー数でチームを作成します。マッチング内のすべてのチームのサイズを均等にする場合は、そのためのルールを作成できます。`EqualTeamSizes` ルールの例に関するトピックは、「FlexMatch ルールセットの例」のトピックを参照してください。

## プレイヤーマッチングのルール設定

マッチングの承諾についてプレイヤーを評価する一連のルールステートメントを作成します。ルールにより、個別のプレイヤー、チーム、またはマッチング全体の要件が設定される場合があります。Amazon GameLift がマッチングリクエストを処理する際には、使用可能なプレイヤーのプールで最も古いプレイヤーから開始し、そのプレイヤーを中心にマッチを構築します。FlexMatch ルール作成の詳細なヘルプについては、「FlexMatch ルールタイプ」を参照してください。

- name (必須) – ルールセット内のルールをユニークに識別する意味のある名前。ルール名は、このルールに関連するアクティビティを追跡するイベントログとメトリクスでも参照されます。
- [description](説明 )(オプション) この要素を使用して自由形式のテキストの説明をアタッチします。
- [type](タイプ) (必須) タイプ要素は、ルールを処理する際に使用するオペレーションを識別します。各ルールタイプには一連の追加プロパティが必要です。有効なルールタイプとプロパティのリストについては、「FlexMatch ルール言語」を参照してください。
- ルールタイププロパティ (必須の場合があります) 定義するルールの種類に応じて、特定のルールプロパティの設定が必要になる場合があります。プロパティと FlexMatch プロパティ表現言語の使用方法については、「FlexMatch ルール言語」の詳細はこちら。

## 時間の経過による要件の許可

拡張により、 FlexMatch がマッチを見つけることができない場合に、時間の経過とともにルール基準を緩和できます。この機能により、FlexMatch では完璧なマッチが得られない場合でも最適な結果が得られます。拡張によりルールを緩和すると、対戦可能なプレイヤーのプールが徐々に拡大されます。

拡張は、不完全なマッチの最新のチケットの経過時間が拡張待機時間と一致する場合に開始されます。FlexMatch が新しいチケットをマッチに追加すると、拡張待機時間クロックがリセットされることがあります。拡張がルールセットの `algorithm` セクションで開始する方法をカスタマイズできます。

以下に、試合に必要な最低スキルレベルが徐々に上昇する拡張の例を挙げます。ルールセットは SkillDelta という名前の距離ルールステートメントを使用して、試合内のすべてのプレイヤーが 5 スキルレベル以内であることを要求します。15 秒間新しいマッチが作成されない場合は、この拡張はスキルレベルの差 10 を探し、10 秒後に 20 の差を探します。

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

自動バックフィルが有効なマッチメーカーで使用されている場合、プレイヤーカウントの要件を急に緩和しないでください。新しいゲームセッションを起動し、自動バックフィルを開始するには数秒かかります。より良い方法は、自動バックフィルがゲームで開始された後に拡張待機時間を開始することです。拡張タイミングはチームの構成によって異なるため、ゲームに最適な拡張戦略を見つけるためにテストを実施します。

## ラージ対戦ルールセットの設計

41 ~ 200 人のプレイヤーを許可するマッチングがルールセットで作成される場合、ルールセットの設定を調整する必要があります。これらの調整により、対戦アルゴリズムが最適化され、プレーヤーの待機時間を短くしながら、実行可能なラージな試合を構築できます。その結果、ラージな対戦ルールセットでは、時間のかかるカスタムルールを、一般的なマッチメーキング優先度に最適化されたスタンダード ソリューションに置き換えます。

ラージな対戦に対してルールセットを最適化する必要があるかどうかを判断する方法は次のとおりです。

1. ルールセットで定義された各チームについて、最大プレイヤー数の値を取得します、
2. [maxPlayer](最大プレイヤー数)値のすべてを追加 この設定が 40 を超えている場合は、ラージ対戦ルール設定を保持しています。

ラージ対戦に対してルールセットを最適化するには、次のように調整します。ラージ対戦ルールセットのスキーマについては、「大規模対戦用のルールセットスキーマ」およびルールセットの例「例 7: ラージ試合を作成する」 を参照してください。

### ラージ対戦アルゴリズムのカスタマイズ

アルゴリズムコンポーネントがまだ存在しない場合は、アルゴリズムコンポーネントをルールセットに追加します。以下のパラメータを設定します。

- `strategy` (必須)`strategy`プロパティを「バランス」に設定します。この設定は、FlexMatch をトリガーして、指定されたプレイヤー属性 (`balancedAttribute` プロパティで定義) に基づいて最適なチームバランスを見つけるために、追加のマッチ後チェックを実行します。バランスの取れた戦略は、均等に対戦しているチームを構築するためのカスタムルールの必要性を置き換えます。
- `balancedAttribute` (必須)— 試合中のチームのバランスをとるときに使用するプレイヤー属性を識別します。この属性は、数値データ型 (倍精度または整数) でなければなりません。たとえば、プレイヤースキルのバランスをとることを選択した場合、FlexMatch はすべてのチームに可能な限り均等にマッチする合計スキルレベルを持つようにプレイヤーを割り当てようとします。ルールセットのプレイヤー属性で、必ずバランシング属性を必ず宣言してください。

- batchingPreference (オプション)— プレイヤーにとって可能な限り低いレイテンシー マッチを形成する際にどの程度の重点を置きたいかを選択します。この設定は、試合を構築する前に対戦チケットをソートする方法に影響します。オプションには以下が含まれます。
  - 最大母集団 FlexMatch では、共通する少なくとも 1 つのロケーションで許容されるレイテンシー値を持つプール内のすべてのチケットを使用してマッチを許可します。その結果、潜在的なチケットプールはラージになる傾向があり、試合をより迅速に埋めることが容易になります。プレイヤーは、許容できるが、必ずしも最適ではないレイテンシーでゲームに参加することができます。batchingPreference プロパティが設定されていない場合、この動作は、strategy が「バランス」に設定されているときにデフォルトになります。
  - 最速のロケーション。FlexMatch は、最も低いレイテンシー値を報告する場所に基づいて、プール内のすべてのチケットを事前にソートします。その結果、同じロケーションでレイテンシーが低いプレイヤーとのマッチが形成される傾向があります。同時に、各試合の潜在的なチケットプールが小さくなり、試合の完了に必要な時間が長くなります。また、レイテンシーに高い優先順位が設定されるため、マッチ中のプレイヤーは、バランシング属性に関して、より差が広がる可能性があります。

以下の例では、次のように動作するように対戦アルゴリズムを設定します。(1) チケットプールを事前にソートして、許容レイテンシー値を持つリージョン別にチケットをグループ化し、(2) ソートされた対戦チケットのバッチを形成し、(3) バッチ内に試合チケットを作成して、プレイヤースキルを平均化できるようにチームのバランスを取ります。

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

プレイヤー属性の宣言

少なくとも、ルールセットのアルゴリズムでバランシング属性として使用されるプレイヤー属性を宣言する必要があります。この属性は、マッチメーキングリクエストの各プレイヤーに対して含める必要があります。プレイヤー属性にはデフォルト値を指定できますが、プレイヤー固有の値を提供した場合に、属性のバランシングが最適に機能します。

### チームの定義

チームサイズと構造を定義するプロセスは小規模の対戦と同様ですが、 FlexMatch がチームを満たす方法は異なります。これは、部分的に満たされた場合の試合に影響します。これに応じて、チームの最小サイズを調整できます。

プレイヤーをチームに割り当てる際に、 FlexMatch は以下のルールを使用します。1: 最小プレイヤー要件に到達していないチームを探す。2: これらのチームのうち、空きスロットが最も多いチームを探す。

複数の均等なサイズのチームを定義するマッチングでは、いっぱいになるまでプレイヤーが順に各チームに追加されます。その結果、マッチングがいっぱいでなくても、マッチングのチームのプレイヤー数は、常にほぼ同数になります。現時点では、大規模マッチングでチームサイズを強制的に均等にすることはできません。非対称のチームサイズのマッチングの場合、プロセスはもう少し複雑です。この場合、プレイヤーは空きスロットが最も多い最大のチームに最初に割り当てられます。次に、空きスロットの数がすべてのチーム間でより均等に分配されるにつれて、プレイヤーはより小さなチームに追加され始めます。

たとえば、3 つのチームで構成されるルールセットがあるとします。赤チームと青チームはどちらも`maxPlayers=10`、`minPlayers=5` に設定されます。グリーンチームは`maxPlayers=3`、`minPlayers=2` に設定されています。塗りつぶし順序は次のとおりです。

1. どのチームも到達していません`minPlayers`。赤チームと青チームには 10 個の空きスロットがあり、緑チームには 3 個の空きスロットがあります。最初の 10 人のプレイヤー (5 人ごと) は、赤チームと青チームに割り当てられます。両方のチームが達しました `minPlayers`。
2. 緑チームはまだ達していません `minPlayers`。次の 2 人のプレイヤーが緑チームに割り当てられます。グリーンチームが現在達しました `minPlayers`。
3. これですべてのチームが `minPlayers` で、オープンスロット数に基づいて追加のプレイヤーが割り当てられてるようになりました。赤チームと青チームには それぞれ5個の空きスロットがあり、緑チームには1個の空きスロットがあります。次の8人のプレイヤー は、赤チームと青チームに(それぞれ4人) 割り当てられます。すべてのチームに 1 つのオープンスロットがあります。
4. 残りの 3 個のプレイヤースロットは、順不同でチームに (1 個ずつ) が割り当てられます。

### ラージマッチのルールを設定する

ラージな対戦のマッチメーキングは、主にバランシング戦略とレイテンシーのバッチ最適化に依存します。ほとんどのカスタムルールは使用できません。ただし、以下の種類のルールを組み込むこともできます。

- プレイヤーのレイテンシーに厳しい制限を設定するルール。latency ルールタイプはプロパティ maxLatency と一緒に使用してください。レイテンシールール リファレンスを参照してください。最大プレイヤーレイテンシーを 200 ミリ秒に設定する例を次に示します。

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

- 指定したプレーヤー属性の近さに基づいてプレーヤーをバッチ処理するルール。これは、均等にマッチしたチームを構築することに重点を置くラージマッチアルゴリズムの一部としてバランス属性を定義することとは異なります。このルールは、ビギナースキルや エキスパートスキルなど、指定された属性値の類似性に基づいてマッチメーキングチケットを一括処理します。そのため、特定の属性について密接に整合しているプレイヤー同士がマッチすることになる傾向があります。batchDistance ルールタイプを使用し、数値ベースの属性を特定し、許可する最も広い範囲を指定します。バッチ距離ルール リファレンスを参照してください。マッチのプレイヤー同士のスキルレベルが 1 つ以内であることを求める例を以下に示します。

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

ラージな対戦要件の緩和

小規模なマッチングの場合と同様に、マッチングが不可能な場合に時間の経過ともに要件を緩和する拡張を使用できます。ラージな対戦では、レイテンシー ルールを緩和するか、チームプレイヤーカウントを緩和するか選択できます。

ラージな対戦に自動マッチングバックフィルを使用している場合は、チームプレイヤーカウントを急に緩和しないでください。FlexMatch は、ゲームセッションが開始された後にのみバックフィルリクエストを生成します。この動作は、試合が作成された後数秒間は発生しないことがあります。その間、 FlexMatch は部分的に満たされた複数の新しいゲームセッションを作成します。これは特にプレイヤーカウントルールを低くした場合に発生します。その結果、必要以上の数のゲームセッションが作成され、ゲームセッション間のプレイヤーがまばらになります。ベストプラクティスは、最初のステップとして、ゲームセッションを開始するのに十分な、プレイヤー数の拡張により長い時間を設定します。バックフィルリクエストでは大規模マッチングにより高い優先度が与えられるため、着信

プレイヤーは新しいゲームが開始される前に既存のゲームのスロットに配置されます。ゲームに対して最適な待機時間を見つけるために、実験が必要になる場合があります。

黄色チームの待機時間を最初よりも長くして、段階的にプレイヤーカウントを低くする例を次に示します。ルールセット内の待機時間は絶対値であり、複合されないことに注意してください。したがって、最初の拡大が 5 秒で発生し、2 番目の拡張はその 5 秒後から10 秒ごとに発生します。

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

## マッチメーキングルールセットの作成

Amazon マッチメーカーの GameLift FlexMatch マッチメーキングルールセットを作成する前に、ルールセット構文 を確認することをお勧めします。Amazon GameLift コンソールまたは AWS Command Line Interface (AWS CLI) を使用してルールセットを作成した後は、変更できません。

AWS リージョンに設定できるルールセットの最大数にはサービスクォータがあるため、未使用のルールセットは削除することをお勧めします。

関連トピック

- FlexMatch ルールセットの設計
- FlexMatch ルールセットの例
- FlexMatch ルール言語

Console

ルールセットを作成する

1. https://console.aws.amazon.com/gamelift/ で Amazon GameLift コンソールを開きます。
2. ルールセットの作成先の AWS リージョンに切り替えます。ルールセットは、ルールセットを使用するマッチメーキング設定と同じリージョンに定義します。

3. ナビゲーションペインで、FlexMatch、マッチメーキングルールセットを選択します。
4. [マッチメーキングルールセット] ページで、[ルールセットを作成] を選択します。
5. [マッチメーキングルールセットの作成] ページで、次の操作を行います。
   a. [ルールセット設定] の [名前] に、リスト、イベント、メトリクステーブルで識別できる一意のわかりやすい名前を入力します。
   b. [ルールセット] には、JSON 形式のルールセットを入力します。ルール指定に関する情報は、「FlexMatch ルールセットの設計」を参照してください。FlexMatch ルールセットの例 からルールセット例を使用することもできます。
   c. [検証] をクリックし、ルールセットの構文が正しいことを検証します。ルールセットは、作成した後に編集できないため、ルールセットを検証するようお勧めします。
   d. (オプション) [タグ] に、リソースの管理と追跡に役立つタグを追加します。AWS
6. [作成] を選択します。正常に作成されたら、そのルールセットをマッチメーカーで使用できます。

AWS CLI

ルールセットを作成する

コマンドラインウィンドウを開き、 コマンドを使用しますcreate-matchmaking-rule-set。

このコマンド例では、1 つのチームをセットアップする簡単なマッチメーキングルールセットを作成します。ルールセットは、それを使用するマッチメーキング設定と同じ AWS リージョンで必ず作成してください。

```
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}]}'
```

作成リクエストが成功すると、Amazon は指定した設定を含む MatchmakingRuleSet オブジェクト GameLift を返します。これでマッチメーカーで新しいルールセットを使用できるようになりました。

Console

ルールセットを削除する

1. https://console.aws.amazon.com/gamelift/ で Amazon GameLift コンソールを開きます。
2. ルールセットを作成したリージョンに切り替えます。
3. ナビゲーションペインで、FlexMatch、マッチメーキングルールセットを選択します。
4. [マッチメーキングルールセット] ページで、削除するルールセットを選択し、[削除] を選択します。
5. [ルールセットの削除] のダイアログボックスで、[削除] を選択して確認します。

> **Note**
>
> マッチメーキング設定でルールセットが使用されている場合、Amazon GameLift はエラーメッセージを表示します (ルールセット は削除できません )。この場合は、別のルールセットを使用するようにマッチメーキング設定を変更してから、もう一度試してください。ルールセットを使用しているマッチメーキング設定を見つけるには、ルールセット名を選択してその詳細ページを表示します。

AWS CLI

ルールセットを削除する

コマンドラインウィンドウを開き、 コマンドを使用してマッチメーキングルールセットdelete-matchmaking-rule-setを削除します。

マッチメーキング設定でルールセットが使用されている場合、Amazon はエラーメッセージ GameLift を返します。この場合は、別のルールセットを使用するようにマッチメーキング設定を変更してから、もう一度試してください。ルールセットを使用しているマッチメーキング設定のリストを取得するには、 コマンドを使用してルールセット名describe-matchmaking-configurationsを指定します。

このコマンド例では、最初にマッチメーキングルールセットの使用状況を確認し、次にルールセットを削除します。

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

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

## FlexMatch ルールセットの例

FlexMatch ルールセットは、さまざまなマッチメーキングシナリオに対応できます。次の例は、FlexMatch 構成構造とプロパティ式言語に準拠しています。これらのルールセット全体をコピーするか、必要に応じてコンポーネントを選択します。

FlexMatch ルールとルールセットの使用の詳細については、以下のトピックを参照してください。

- FlexMatch ルールセットの構築
- FlexMatch ルールセットの設計
- FlexMatch ルールセットスキーマ
- FlexMatch ルール言語

> **Note**
>
> 複数のプレイヤーが含まれるマッチメーキングチケットを評価する場合は、リクエスト内のすべてのプレイヤーがマッチング要件を満たす必要があります。

### 例 1: プレイヤーが均等にマッチングされる 2 つのチームを作成する

この例では、プレイヤーが均等にマッチングされる 2 つのチームを設定する手順を示します。

- プレイヤーのチームを 2 つ作成します。
  - 各チームに 4〜8 名のプレイヤーを含めます。
  - 最終的に両チームのプレイヤー数は同じにする必要があります。
- プレイヤーのスキルレベルを含めます (指定しない場合、デフォルトの 10 が使用されます)。
- スキルレベルが類似するプレイヤーを選択します。両チームのプレイヤーの平均スキル差は 10 ポイント以内とします。
- すぐにマッチングが満たされない場合は、妥当な時間内にマッチングが完了するようにプレイヤーのスキル要件を緩和します。
  - 5 秒後に、検索範囲を広げて平均スキル差が 50 ポイント以内のプレイヤーを対象にします。

- 15 秒後に、検索範囲を広げて平均スキル差が 100 ポイント以内のプレイヤーを対象にします。

このルールセットの使用に関する注意事項

- この例では、チームのサイズが 4 ～ 8 プレイヤーの任意のチームを対象にしています (ただし、両チームのサイズは同じにする必要があります)。チームのサイズが有効な範囲内である場合、マッチメーカーはできる限り最大数のプレイヤーをマッチングします。
- FairTeamSkill ルールでは、プレイヤーのスキルに基づいてチームを均等にマッチングします。新たな見込みプレイヤーごとにこのルールを評価するために、 FlexMatch は暫定的にチームにプレーヤーを追加し、平均を計算します。ルールが失敗すると、プレイヤー候補はマッチングに追加されません。
- 両方のチームは同一の構造を持っているため、1 つのチーム定義だけを作成し、チーム数を "2" に設定できます。このシナリオでは、チームを "aliens" と名付けた場合、チームには "aliens_1" と "aliens_2" という名前が割り当てられます。

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

## 例 2: 不均等なチーム (ハンター対モンスター) を作成する

この例は、プレイヤーのグループが単一のモンスターをハントするゲームモードを示しています。プレイヤーは、ハンターまたはモンスターのロールを選択します。ハンターは、敵対するモンスターの最小スキルレベルを指定します。ハンターチームの最小サイズは、マッチングを達成するために徐々に緩和できます。このシナリオでは、以下の手順に従います。

- 正確に 5 名のハンターで構成される 1 つのチームを作成します。
- 正確に 1 匹のモンスターで構成される別のチームを作成します。
- 以下のプレイヤー属性を含めます。
  - プレイヤーのスキルレベル (指定しない場合、デフォルトの 10 が使用されます)。
  - プレイヤーが希望するモンスターのスキルレベル (指定しない場合、デフォルトの 10 が使用されます)。

  - プレイヤーがモンスターのロールを希望するかどうか (指定しない場合、デフォルトで 0 または false になります)。
- 以下の条件に基づいてモンスターとなるプレイヤーを選択します。
  - プレイヤーはモンスターのロールをリクエストする必要があります。
  - プレイヤーは、ハンターチームに既に追加されているプレイヤーが希望する最高のスキルレベルを達成済みであるか、超えている必要があります。
- 以下の条件に基づいてハンターチームに属するプレイヤーを選択します。
  - モンスターのロールをリクエストしたプレイヤーは、ハンターチームに参加できません。
  - モンスターのロールが既に埋まっている場合、プレイヤーが希望するモンスターのスキルレベルは、モンスター候補のスキルより低くなければなりません。
- すぐにマッチングが達成されない場合は、以下のようにハンターチームの最小サイズを緩和します。
  - 30 秒後に、ハンターチームのプレイヤー 4 名のみでゲームを開始することを許可します。
  - 60 秒後に、ハンターチームのプレイヤー 3 名のみでゲームを開始することを許可します。

このルールセットの使用に関する注意事項

- ハンターとモンスターに 2 つの異なるチームを使用することで、さまざまな条件のセットに基づいてメンバーシップを評価できます。

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

```
}
```

## 例 3: チームレベル要件とレイテンシーの制限を設定する

この例は、プレイヤーチームのセットアップ方法と、各プレイヤーの代わりに各チームに一連のルールセットを適用する方法を示しています。3 つの均等にマッチングされたチームを作成するための 1 つの定義を使用します。また、すべてのプレイヤーの最大レイテンシーを設定します。レイテンシーの最大値は、マッチングを達成するために徐々に緩和できます。この例では、以下の手順を開始します。

- プレイヤーのチームを 3 つ作成します。
  - 各チームに 3〜5 名のプレイヤーを含めます。
  - 各チームの最終的なプレイヤー数は同数またはほぼ同数 (差は 1 以内) にする必要があります。
- 以下のプレイヤー属性を含めます。
  - プレイヤーのスキルレベル (指定しない場合、デフォルトの 10 が使用されます)。
  - プレイヤーのキャラクターロール (指定しない場合、デフォルトの「農民」が使用されます)。
- マッチングのスキルレベルが類似するプレイヤーを選択します。
  - 各チームのプレイヤーの平均スキル差は 10 ポイント以内とします。
- チームの「医者」キャラクターを以下の数に制限します。
  - マッチング全体の医者の最大数は 5 とします。
- 50 ミリ秒以下のレイテンシーを報告したプレイヤーのみにマッチングします。
- すぐにマッチングが達成されない場合は、以下のようにプレイヤーのレイテンシー要件を緩和します。
  - 10 秒後に、プレイヤーのレイテンシー値として最大 100 ミリ秒まで許可します。
  - 20 秒後に、プレイヤーのレイテンシー値として最大 150 ミリ秒まで許可します。

このルールセットの使用に関する注意事項

- このルールセットでは、プレイヤーのスキルに基づいてチームを均等にマッチングします。`FairTeamSkill` ルールを評価するために、 は潜在的なプレイヤーをチームに FlexMatch 暫定的に追加し、チーム内のプレイヤーの平均スキルを計算します。次に、これを両方のチームのプレイヤー平均スキルと比較します。ルールが失敗すると、プレイヤー候補はマッチングに追加されません。

- チームレベルおよびマッチングレベルの要件 (医者の総数) は、収集ルールを通じて達成されます。このルールタイプでは、すべてのプレイヤーのキャラクター属性のリストを、最大数に照らしてチェックします。すべてのチームのすべてのプレイヤーのリストを作成するには、flatten を使用します。
- レイテンシーに基づいて評価する場合は、以下の点に注意してください。
  - レイテンシーデータは、Player オブジェクトの一部としてマッチメーキングリクエストで提供されます。これは属性ではないため、属性としてリストする必要はありません。
  - マッチメーカーは、リージョン別にレイテンシーを評価します。レイテンシーが最大数を超えるすべてのリージョンは無視されます。プレイヤーがマッチングで承諾されるためには、レイテンシーが最大値未満のリージョンが少なくとも 1 つ必要です。
  - マッチメーキングリクエストが 1 人または複数のプレイヤーのレイテンシデータを省略した場合、そのリクエストはすべてのマッチで拒否されます。

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

## 例 4: 明示的な並べ替えを使用して最適なマッチングを見つける

この例では、3 人ずつのプレイヤーで構成される 2 つのチームでシンプルなマッチングを設定します。明示的な並べ替えルールを使用して、可能な限り最良のマッチングをできるだけ早く見つける方法を示します。これらのルールでは、すべてのアクティブなマッチメーキングチケットを事前に並べ替え、特定のキーとなる要件に基づいて最適な対戦を作成します。この例は、以下の手順に従って実装します。

- プレイヤーのチームを 2 つ作成します。
- 各チームを正確に 3 人のプレイヤーで構成します。
- 以下のプレイヤー属性を含めます。
  - 経験レベル (指定しない場合、デフォルトで 50 が使用されます)。
  - 優先するゲームモード (複数の値をリスト可能) (指定しない場合、デフォルトで「クープ」と「デスマッチ」が使用されます)。
  - 優先するゲームマップ (マップ名と優先重み付けを含む) (指定しない場合、デフォルトで重み 100 の `"defaultMap"` が使用されます)。
- 事前並べ替えを設定します。
  - アンカープレイヤーとして同じゲームマップを優先する度合いに基づいてプレイヤーを並べ替えます。プレイヤーのお気に入りのゲームマップは複数存在することがあるため、この例では優先値を使用しています。
  - 経験レベルがアンカープレイヤーとどれだけ近くマッチングするかに基づいてプレイヤーを並べ替えます。この並べ替えにより、すべてのチーム間ですべてのプレイヤーの経験レベルができるだけ近いものになります。
- すべてのチーム間ですべてのプレイヤーが少なくとも 1 つのゲームモードを共通して選択している必要があります。
- すべてのチーム間ですべてのプレイヤーが少なくとも 1 つのゲームマップを共通して選択している必要があります。

### このルールセットの使用に関する注意事項

- ゲームマップの並べ替えでは、mapPreference 属性値を比較する絶対並べ替えを使用します。これはルールセットの最初のルールであるため、この並べ替えが最初に実行されます。
- 経験の並べ替えでは、アンカープレイヤーのスキルとともに候補プレイヤーのスキルレベルを比較するために、距離の並べ替えが使用されます。

- 並べ替えは、ルールセットで指定された順に実行されます。このシナリオでは、プレイヤーがゲームマップの優先度によって並べ替えられ、さらに経験レベル順に並べ替えられます。

```
{
    "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
    }]
}
```

## 例 5: 複数のプレイヤー属性間の交差を見つける

この例では、収集ルールを使用して、2 つ以上のプレイヤー属性の交差を見つける方法を説明します。コレクションを操作するときは、1 つの属性に対しては `intersection` オペレーションを使用し、複数の属性に対しては `reference_intersection_count` オペレーションを使用できます。

この方法を説明するために、この例ではキャラクターの設定に基づいて、マッチングのプレイヤーを評価します。サンプルゲームはfree-for-all「」スタイルで、試合内のすべてのプレイヤーが対戦相手

です。各プレイヤーは、(1) 自分のキャラクターを選択し、(2) 対戦するキャラクターを選択することが求められます。マッチングの各プレイヤーが、他のすべてのプレイヤーの希望する対戦相手リストに含まれているキャラクターを使用するようにするルールが必要です。

このルールセットの例では、次の特性を持つマッチングについて説明します。

- チーム構造: 5 人のプレイヤーがいる 1 つのチーム
- プレイヤー属性:
  - myCharacter: プレイヤーが選択したキャラクター。
  - preferredOpponents: プレイヤーが対戦したいキャラクターのリスト。
- マッチングルール: 使用中の各キャラクターが各プレイヤーの希望する対戦リストに含まれている場合、マッチング候補は受け入れ可能です。

マッチングルールを実装するため、この例では次のプロパティ値を持つ収集ルールを使用します。

- オペレーション `reference_intersection_count` オペレーションを使用して、測定値の文字列リストがリファレンス値の文字列リストと交差する方法を評価します。
- 測定 `flatten` プロパティ表現を使用して文字列のリストを作成し、各リストに 1 人のプレイヤーの myCharacter 属性値を含めます。
- リファレンス値 `set_intersection` プロパティ表現を使用して、試合の各プレイヤーに共通するすべての preferredOpponents 属性値を含む文字列のリストを作成します。
- 制約 `minCount` を 1 に設定し、各プレイヤーの選択したキャラクター (測定値の文字列のリスト) が、すべてのプレイヤーに共通の 1 人以上の優先される対戦相手 (リファレンス値の文字列) と一致するようにします。
- 拡張 15 秒以内にマッチングが達成されない場合は、最小の交差要件を緩和します。

このルールのプロセスフローは次のようになります。

1. プレイヤーがマッチング候補に追加されます。参照値 (文字列のリスト) が再計算され、新しいプレイヤーの希望の対戦相手リストに交差が含まれるようにします。計測値 (文字列のリスト) が再計算され、新しいプレイヤーの選択されたキャラクターが新しい文字列リストとして追加されます。
2. Amazon は、測定値の各文字列リスト (プレイヤーが選択したキャラクター) が、リファレンス値 (プレイヤーの優先対戦相手) の少なくとも 1 つの文字列と交差する GameLift ことを確認します。

この例では、測定値の各文字列リストには値が 1 つしか含まれないため、交差は 0 または 1 になります。

3. 測定値の文字列リストが参照値の文字列リストと交差しない場合、ルールは失敗し、新しいプレイヤーはマッチング候補から削除されます。
4. マッチングが 15 秒以内に達成されない場合は、対戦相手のマッチング要件を削除し、マッチングの残りのプレイヤースロットを埋めます。

```
{
    "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
```

```
        }]
    }]
}
```

## 例 6: すべてのプレイヤー間の属性の比較

この例では、プレイヤーのグループ間でプレイヤー属性を比較する方法を示します。

このルールセットの例では、次の特性を持つマッチングについて説明します。

- チーム構造: 2 つの単一プレイヤーチーム
- プレイヤー属性:
  - gameMode: プレイヤーによって選択されたゲームのタイプ (指定されていない場合は、デフォルトで「順番ベース」となります)。
  - gameMap: プレイヤーによって選択されたゲーム世界 (指定されない場合は、デフォルトで 1 になります)。
  - キャラクター: プレイヤーによって選択されたキャラクター (デフォルト値がない場合、プレイヤーはキャラクターを指定する必要があります)。
- マッチングルール: マッチングされたプレイヤーは次の要件を満たす必要があります。
  - プレイヤーは同じゲームモードを選択する必要があります。
  - プレイヤーは同じゲームマップを選択する必要があります。
  - 多くのプレイヤーは異なるキャラクターを選択します。

このルールセットの使用に関する注意事項

- この例では、マッチングルールを実装するため、比較ルールを使用してすべてのプレイヤーの属性値を確認します。ゲームモードとマップについては、値が同じことがルールで確認されます。キャラクターについては、値が異なることがルールで確認されます。
- この例では、両方のプレイヤーチームを作成するために数量プロパティを指定して 1 つのプレイヤー定義を使用します。チームには、"player_1" や "player_2" のような名前が割り当てられます。

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

## 例 7: ラージ試合を作成する

この例では、40 人を超えるプレイヤーのマッチングに対するルールセットを設定する方法を示します。ルールセットでチームの maxPlayer 合計カウントが 40 以上であると定義された場合、大規模なマッチングとして処理されます。詳細については、「ラージ対戦ルールセットの設計」を参照してください。

この例のルールセットでは、以下の手順に従ってマッチングが作成されます。

- 最大 200 人、最低 175 人のプレイヤーがいる 1 つのチームを作成します。
- バランシング条件: 類似したスキルレベルに基づいてプレイヤーを選択します。すべてプレイヤーは、マッチングのためにスキルレベルを報告する必要があります。
- バッチ優先設定: マッチングの作成時に、類似したバランシング条件によってプレイヤーをグループ化します。
- レイテンシールール: 最大許容プレイヤーレイテンシーとして 150 ミリ秒を設定します。
- すぐにマッチングが満たされない場合は、妥当な時間内にマッチングを完了するために要件を緩和します。
  - 10 秒後に、プレイヤーが 150 人のチームを受け入れます。
  - 12 秒後に、許容されるレイテンシーの最大値を 200 ミリ秒に引き上げます。
  - 15 秒後に、プレイヤーが 100 人のチームを受け入れます。

このルールセットの使用に関する注意事項

- アルゴリズムは「最大母集団」バッチ優先設定を使用しているため、プレイヤーはまずバランシング要件に基づいて並べ替えられます。その結果、マッチングはより詳細になり、スキルがより類似したプレイヤーが含まれる可能性が高くなります。すべてのプレイヤーは許容されるレイテンシー要件を満たしますが、その場所での最大限のレイテンシーを取得できない可能性もあります。
- ルールセットで使用されるこのアルゴリズム戦略は、「最大母集団」がデフォルト設定です。デフォルト設定を使用するには、設定を省略することもできます。
- マッチングバックフィルを有効にしている場合は、プレイヤーカウント要件を急に緩和しないでください。緩和が速すぎると、部分的に満たされたゲームセッションが大量に生成される可能性があります。詳細については、「ラージな対戦要件の緩和」を参照してください。

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

## 例 8: 複数チームのラージ試合を作成する

この例は、プレイヤーが 40 人を超える複数チームのマッチング用にルールをセットセットアップする方法を示しています。この例は、1 つの定義を持つ複数の同じチームを作成する方法と、マッチング作成で非対称サイズのチームを満たす方法を示しています。

この例のルールセットでは、以下の手順に従ってマッチングが作成されます。

- 最大 15 人のプレイヤーがいる同一の「ハンター」チームを 10 個と、厳密に 5 人のプレイヤーがいる「モンスター」チームを 1 個作成します。
- バランシング要件: モンスターを倒した数を基準にプレイヤーを選択します。プレイヤーが倒した数を報告しない場合は、デフォルト値の 5 を使用します。
- バッチ優先設定: 最短のプレイヤーレイテンシーが報告されているリージョンを基準に、プレイヤーをグループ化します。
- レイテンシールール: 許容されるプレイヤーレイテンシーの最大値を 200 ミリ秒に設定します。
- すぐにマッチングが満たされない場合は、妥当な時間内にマッチングを完了するために要件を緩和します。
  - 15 秒後に、10 人のプレイヤーチームを受け入れます。
  - 20 秒後に、8 人のプレイヤーチームを受け入れます。

このルールセットの使用に関する注意事項

- このルールセットは、潜在的に最大 155 人のプレイヤーを保持できるチームを定義します。これはラージ試合となります。( 10×15ハンター　+　5　モンスター　=　155 )
- アルゴリズムは「最短リージョン」バッチ優先設定を使用しているため、プレイヤーはより高い (許容範囲内の) レイテンシーを報告しているリージョンよりも、より速いレイテンシーを報告しているリージョンに配置される傾向があります。同時に、マッチングのプレイヤーはより少数になり、バランシング条件 (モンスタースキルの数) はより広範囲になる可能性があります。
- 拡張は、複数チーム (数量 > 1) に対して定義された場合、定義を作成したすべてのチームに適用されます。したがって、ハンターチームの最小プレイヤー設定を緩和することによって、10 個すべてのハンターチームが同様に影響を受けます。
- このルールセットはプレイヤーレイテンシーを最小にするために最適化されているため、このレイテンシールールは許容される接続オプションを持たないプレイヤーを除外するキャッチオールとして機能します。この要件を緩和する必要はありません。
- 拡張が有効になる前に、 がこのルールセットの一致 FlexMatch を満たす方法は次のとおりです。

- どのチームも minPlayers カウントにまだ達していません。ハンターチームには 15 個の空きスロットがあり、モンスターチームには 5 つの空きスロットがあります。
  - 最初の 100 人のプレイヤーが (10 人ずつ) 10 個のハンターチームに割り当てられます。
  - 次の 22 名のプレイヤーは順番に (2 人ずつ) ハンターチームとモンスターチームに割り当てられます。
- ハンターチームはそれぞれ minPlayers カウントである 12 人のプレイヤーに達しました。モンスターチームには 2 人のプレイヤーがいて、minPlayers カウントには達していません。
  - 次の 3 人のプレイヤーがモンスターチームに割り当てられます。
- すべてのチームが minPlayers カウントに達しました。ハンターチームにはそれぞれ 3 つの空きスロットがあります。モンスターチームのスロットがいっぱいになりました。
  - 最後の 30 人のプレイヤーが順にハンターチームに割り当てられ、すべてのハンターチームがほぼ同じサイズ (+/- 1 人のプレイヤー) になります。
- このルールセットを使用して作成されたマッチングに対してバックフィルが有効になっている場合、プレイヤーカウント要件を急に緩和しないでください。緩和が速すぎると、部分的に満たされたゲームセッションが大量に作成される可能性があります。詳細については、「ラージな対戦要件の緩和」を参照してください。

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

## 例 9: 類似の属性を持つプレイヤーとのラージ試合を作成する

この例では、batchDistance を使用して 2 つのチームとの試合にルールセットを設定する方法を示します。これらの例では:

- SimilarLeagueルールにより、試合内のすべてのプレイヤーがleague 2人以内の他のプレイヤーを保持します。
- SimilarSkillルールにより、試合内のすべてのプレイヤーがskill 10人以内の他のプレイヤーを保持します。プレイヤーが 10 秒待っている場合、距離は20に拡大されます。プレイヤーが 20 秒待っている場合、距離は40に拡大されます。
- SameMapルールにより、試合内のすべてのプレイヤーが同じことをリクエストしていることが保証されますmap。
- SameModeルールにより、試合内のすべてのプレイヤーが同じことをリクエストしていることが保証されます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
        }]
    }]
}
```

## 例 10: 複合ルールを使用して、類似の属性または類似のセレクションを持つプレイヤーとのマッチを作成する

この例では、compound を使用して 2 つのチームとの試合にルールセットを設定する方法を示します。これらの例では:

- SimilarLeagueDistanceルールにより、試合内のすべてのプレイヤーがleague 2人以内の他のプレイヤーを保持します。
- SimilarSkillDistanceルールにより、試合内のすべてのプレイヤーがskill 10人以内の他のプレイヤーを保持します。プレイヤーが 10 秒待っている場合、距離は20に拡大されます。プレイヤーが 20 秒待っている場合、距離は40に拡大されます。
- SameMapComparisonルールにより、試合内のすべてのプレイヤーが同じことをリクエストしていることが保証されますmap。
- SameModeComparisonルールにより、試合内のすべてのプレイヤーが同じことをリクエストしていることが保証されますmode。
- CompoundRuleMatchmaker ルールにより、以下の条件の内 1 つが true である場合に、マッチを保証します。
  - マッチ内のプレイヤーは同じ map と同じ mode リクエストしています。
  - マッチに参加したプレイヤーには、同等の skill および league 属性があります。

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

## 例 11: プレイヤーのブロックリストを使用するルールを作成する

この例は、プレーヤーが他の特定のプレーヤーとマッチングされないようにするルールセットを示しています。プレイヤーはブロックリストを作成できます。ブロックリストは、マッチのプレイヤー選択時にマッチメーカーが評価します。ブロックリストまたは回避リスト機能の追加に関する詳しいガイダンスについては、「ゲームブログの AWS」を参照してください。

この例では、以下の手順を開始します。

- 正確に 5 人のプレイヤーで構成される 2 つのチームを作成します。
- プレイヤー ID (最大 100 個) のリストであるプレイヤーのブロックリストを渡します。
- 全プレイヤーを各プレイヤーのブロックリストと比較し、ブロックされたプレイヤー ID が見つかった場合はマッチを拒否します。

### このルールセットの使用に関する注意事項

- 新規プレイヤーを評価して提案されたマッチに追加する (または既存のマッチでポジションをバックフィルする) 場合、そのプレイヤーは以下のいずれかの理由で拒否されることがあります。
  - その新規プレイヤーが、すでにマッチに選ばれているプレイヤーのブロックリストに載っている場合。
  - マッチにすでに選ばれているプレイヤーが新しいプレイヤーのブロックリストに載っている場合。

- このように、このルールセットでは、ブロックリストにあるどのプレイヤーともプレイヤーをマッチングさせません。ルール拡張を追加して maxCount 値を増やすことで、この要件をプリファレンス (「回避」リストとも呼ばれる) に変更できます。

```
{
    "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
    }]
}
```

# マッチメーキング設定の作成

Amazon GameLift FlexMatch マッチメーカーをセットアップしてマッチメーキングプロセスをリクエストするには、マッチメーキング設定を作成します。Amazon GameLift コンソールまたは AWS Command Line Interface (AWS CLI) を使用する マッチメーカーの作成方法の詳細については、「FlexMatch マッチメーカーの設計」を参照してください。

# Amazon GameLift ホスティングのマッチメーカーを作成する

マッチメーキング設定を作成する前に、このマッチメーキングを使用するマッチメーカーで使用するルールセットを作成し、Amazon GameLift ゲームセッションキューを作成しておきます。

Console

1. [Amazon GameLift コンソール] のナビゲーションペインで、[マッチメーキング設定] を選択します。
2. マッチメーカーの作成先の AWS リージョンに切り替えます。
3. [マッチメーキング設定] ページで [マッチメーキング設定を作成] を選択します。
4. [設定の詳細を定義] ページの [マッチメーキング設定の詳細] で、次の操作を行います。

   a. [名前] マッチメーカーをリストとメトリクスで簡単に識別するのに役立つ名前を入力します。マッチメーカー名は、ロケーション内で一意である必要があります。マッチメーキングリクエストでは、どのマッチメーカーを使用するかを、その名前とリージョンで判断します。

   b. (オプション) [説明] では、マッチメーカーを識別するのに役立つ説明を追加します。

   c. [ルールセット] では、このマッチメーカーで使用するルールセットをリストから選択します。現在のロケーションで作成済みのすべてのルールセットが一覧表示されます。

   d. [FlexMatch モード] では、Amazon GameLift マネージドホスティングの [マネージド] を選択します。このモードでは、指定されたゲームセッションキューに成功したマッチングを渡すよう FlexMatch に要求します。

   e. [AWS リージョン] で、マッチメーカーで使用するゲームセッションキューを設定したリージョンを選択します。

   f. [キュー] では、このマッチメーカーで使用するゲームセッションキューを選択します。
5. [Next] (次へ) をクリックします。
6. [設定を構成] ページの [マッチメーキング設定] で、次の操作を行います。

   a. [リクエストのタイムアウト] では、マッチメーカーが各リクエストでマッチングを完了するまでの最大時間を秒単位で設定します。FlexMatch は、この時間を超えるマッチメーキングリクエストをキャンセルします。

   b. [バックフィルモード] では、マッチバックフィルを処理するモードを選択します。

      - 自動バックフィル機能を有効にするには、[自動] を選択します。

- 独自のバックフィルリクエスト管理を作成する場合や、バックフィル機能を使用しない場合は、[手動] を選択します。

c. (オプション) [追加プレイヤー数] では、マッチ中に空けておくプレイヤースロットの数を設定します。FlexMatch はこれらのスロットをプレイヤーで埋めることができます。

d. (オプション) [マッチングの承諾オプション] の [承諾が必要] では、マッチング案の各プレイヤーに対してマッチングへの参加を積極的に承諾することを要求する場合、[必須] を選択します。このオプションを選択した場合は、[承諾のタイムアウト] で、マッチングをキャンセルするまでにマッチメーカーがプレイヤーの承諾を待機する時間を指定します。

7. (オプション) [イベント通知の設定] で、次の操作を行います。

a. (オプション) [SNS トピック] で、マッチメーキングイベント通知を受信するための Amazon Simple Notification Service (Amazon SNS) トピックを選択します。SNS トピックをまだ設定していない場合は、後でマッチメーキング設定を編集して、これを追加できます。詳細については、「FlexMatch イベント通知をセットアップする」を参照してください。

b. (オプション) [カスタムイベントデータ] では、イベントメッセージングで、このマッチメーカーと関連付けるカスタムデータを入力します。FlexMatch は、マッチメーカーと関連付けられたすべてのイベントにこのデータを含めます。

8. (オプション) [追加のゲームデータ] を展開し、次の操作を実行します。

a. (オプション) [ゲームセッションデータ] では、このマッチメーキング設定を使用して行われたマッチで開始される新しいゲームセッションに FlexMatch が配信する追加のゲーム関連情報を提供します。

b. (オプション) [ゲームプロパティ] には、新しいゲームセッションに関する情報を含むキーと値のペアのプロパティを追加します。

9. (オプション) [タグ] に、リソースの管理と追跡に役立つタグを追加します。AWS

10. [Next] (次へ) をクリックします。

11. [確認および作成] ページで、選択内容を認し、[データストアを作成] を選択します。マッチメーカーが正常に作成されると、マッチメーキングリクエストをすぐに受け入れ可能になります。

AWS CLI

AWS CLI でマッチメーキング設定を作成するには、コマンドラインウィンドウを開き、create-matchmaking-configuration コマンドを使って新しいマッチメーカーを定義します。

このコマンド例では、プレイヤーの承諾を要求して自動バックフィルを有効にする、新しいマッチメーキング設定を作成します。また、FlexMatch 用に 2 つのプレイヤースロットを予約して、ゲーム セッションデータを提供します。

```
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"
```

マッチメーキング設定の作成リクエストが成功すると、マッチメーカーにリクエストした設定が含まれている MatchmakingConfiguration オブジェクトが Amazon GameLift から返されます。新しいマッチメーカーは、マッチメーキングリクエストを受け入れ可能になります。

## スタンドアロンの FlexMatch のマッチメーカーの作成

マッチメーキング設定を作成する前に、マッチメーカーで使用するルールセットを作成して、マッチメーカーと使用します。

Console

1. https://console.aws.amazon.com/gamelift/homeで Amazon GameLift コンソールを開きます。。

2. マッチメーカーの作成先の AWS リージョンに切り替えます。FlexMatch マッチメーキング設定をサポートするリージョンのリストは、「マッチメーカーのロケーションを選択する」を参照してください。
3. ナビゲーションペインで、[FlexMatch]、[マッチメーキング設定] を選択します。
4. [マッチメーキング設定] ページで [マッチメーキング設定を作成] を選択します。
5. [設定の詳細を定義] ページの [マッチメーキング設定の詳細] で、次の操作を行います。

   a. [名前] マッチメーカーをリストとメトリクスで簡単に識別するのに役立つ名前を入力します。マッチメーカー名は、ロケーション内で一意である必要があります。マッチメーキングリクエストでは、どのマッチメーカーを使用するかを、その名前とリージョンで判断します。

   b. (オプション) [説明] では、マッチメーカーを識別するのに役立つ説明を追加します。

   c. [ルールセット] では、このマッチメーカーで使用するルールセットをリストから選択します。現在のロケーションで作成済みのすべてのルールセットが一覧表示されます。

   d. [FlexMatch モード] の場合は、[スタンドアロン] を選択します。これは、ゲームに Amazon GameLift 以外のホスティングソリューションで新しいゲームセッションを開始するためのカスタムメカニズムがあることを示しています。

6. [Next] (次へ) をクリックします。
7. [設定を構成] ページの [マッチメーキング設定] で、次の操作を行います。

   a. [リクエストのタイムアウト] では、マッチメーカーが各リクエストでマッチングを完了するまでの最大時間を秒単位で設定します。この時間を超えたマッチメーキングリクエストは拒否されます。

   b. (オプション) [マッチングの承諾オプション] の [承諾が必要] では、マッチング案の各プレイヤーに対してマッチングへの参加を積極的に承諾することを要求する場合、[必須] を選択します。このオプションを選択した場合は、[承諾のタイムアウト] で、マッチングをキャンセルするまでにマッチメーカーがプレイヤーの承諾を待機する時間を指定します。

8. (オプション) [イベント通知の設定] で、次の操作を行います。

   a. (オプション) [SNS トピック] で、マッチメーキングイベント通知を受信するための Amazon SNS トピックを選択します。SNS トピックをまだ設定していない場合は、後でマッチメーキング設定を編集して、これを追加できます。詳細については、「FlexMatch イベント通知をセットアップする」を参照してください。

b. (オプション) [カスタムイベントデータ] では、イベントメッセージングで、このマッチメーカーと関連付けるカスタムデータを入力します。FlexMatch は、マッチメーカーと関連付けられたすべてのイベントにこのデータを含めます。

9. (オプション) [タグ] に、リソースの管理と追跡に役立つタグを追加します。AWS
10. [Next] (次へ) をクリックします。
11. [確認および作成] ページで、選択内容を認し、[データストアを作成] を選択します。マッチメーカーが正常に作成されると、マッチメーキングリクエストをすぐに受け入れ可能になります。

AWS CLI

AWS CLI でマッチメーキング設定を作成するには、コマンドラインウィンドウを開き、create-matchmaking-configuration コマンドを使って新しいマッチメーカーを定義します。

このコマンド例では、プレイヤーの承諾を要求して自動バックフィルを有効にする、新しいマッチメーキング設定を作成します。

```
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"
```

マッチメーキング設定の作成リクエストが成功すると、マッチメーカーにリクエストした設定が含まれている MatchmakingConfiguration オブジェクトが Amazon GameLift から返されます。新しいマッチメーカーは、マッチメーキングリクエストを受け入れ可能になります。

## マッチメーキング設定を編集する

マッチメーキング設定を編集するには、ナビゲーションバーから [マッチメーキング設定] を選択し、編集する設定を選択します。既存の設定のフィールドは、名前以外のどのフィールドでも更新できます。

設定ルールセットを更新する際、有効なマッチメーキングチケットが存在する場合、以下の理由により新しいルールセットは互換性がなくなる可能性があります。

- 新しいチーム名または異なるチーム名、またはチーム数
- 新しいプレイヤー属性
- 既存のプレイヤー属性タイプの変更

ルールセットにこれらの変更を加えるには、更新されたルールセットを使って新しいマッチメーキング設定を作成します。

# FlexMatch イベント通知をセットアップする

イベント通知を使用して、個々のマッチメーキングリクエストのステータスを追跡できます。本番環境のすべてのゲーム、または大量のマッチメーキングアクティビティがある本番前の環境では、イベント通知を使用する必要があります。

イベント通知を設定するためには二つのオプションがあります。

- マッチメーカーに Amazon Simple Notification Service (Amazon SNS) トピックにイベント通知をパブリッシュさせます。
- 自動的にパブリッシュされる Amazon EventBridge イベントとイベント管理ツール一式を使用します。

Amazon GameLift が発行する FlexMatch イベントのリストについては、「FlexMatch マッチメーキングイベント」を参照してください。

## EventBridge イベントをセットアップする

Amazon GameLift はすべてのマッチメーキングイベントを自動的に Amazon EventBridge に発行します。EventBridge を使用すると、イベントを処理するために、ターゲットにイベントをルーティングするルールを設定できます。例えば、「PotentialMatchCreated」イベントを、プレイヤーの承諾を処理する AWS Lambda 関数にルーティングするルールを設定できます。詳細については、「Amazon EventBridge とは」を参照してください。

> **Note**
>
> マッチメーカーの設定時に、通知ターゲットのフィールドは空にするか、または EventBridge と Amazon SNS の両方を使用する場合は、SNS トピックを参照します。

## Amazon SNS トピックを設定する

Amazon GameLift に FlexMatch マッチメーカーが生成するすべてのイベントを Amazon SNS トピックにパブリッシュさせることができます。

Amazon GameLift イベント通知用 SNS トピックを作成するには

1. [Amazon SNS console] (Amazon SNS コンソール) を開きます。
2. ナビゲーションペインで、[トピック] を選択します。
3. [トピック] ページで、[トピックの作成] を選択します。
4. コンソールでトピックを作成します。詳細については、Amazon Simple Notification Service 開発者ガイドの「AWS Management Console を使用してトピックを作成するには」を参照してください。
5. トピックの [詳細] ページで、[編集] を選択します。
6. (オプション) トピックの [編集] ページで [アクセスポリシー] を展開し、次の AWS Identity and Access Management (IAM) ポリシーステートメントの太字の構文を既存のポリシーの末尾に追加します。(ここではわかりやすいようにポリシー全体を示しています。) 独自の SNS トピックと Amazon GameLift のマッチメーキング設定には、Amazon リソースネーム (ARN) の詳細を必ず使用してください。

```
{
  "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. [Save changes] (変更の保存) をクリックします。

## サーバー側の暗号化を使用して SNS トピックをセットアップする

サーバー側の暗号化 (SSE) を使用して、暗号化されたトピックに機密データを保管できます。SSE では、AWS Key Management Service (AWS KMS) で管理されているキーを使用して、Amazon SNS キュー内のメッセージの内容が保護されます。Amazon SNS によるサーバー側の暗号化の詳細については、Amazon Simple Storage Service 開発者ガイドの「保管時の暗号化」を参照してください。

サーバー側の暗号化を使用して SNS トピックを設定する方法については、以下のトピックを確認してください。

- AWS Key Management Service デベロッパーガイドの「キーの作成」。
- Amazon Simple Notification Service 開発者ガイドの「トピックの SSE を有効にする」

KMS キーを作成するときは、次の KMS キーポリシーを使用します。

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

## トピックサブスクリプションを設定して Lambda 関数を呼び出す

Amazon SNS トピックに発行されたイベント通知を使用して Lambda 関数を呼び出すことができます。マッチメーカーの設定時に、必ず通知ターゲットを SNS トピックの ARN に設定します。

次の AWS CloudFormation テンプレートは、`MyFlexMatchEventTopic` という名前の SNS トピックへのサブスクリプションを設定して、`FlexMatchEventHandlerLambdaFunction` という名前の Lambda 関数を呼び出します。このテンプレートは、Amazon GameLift が SNS トピックに書き込むことを許可する IAM アクセス許可ポリシーを作成します。次に、テンプレートは、SNS トピックに Lambda 関数を呼び出すアクセス許可を追加します。

```
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
```

# FlexMatch のゲームの準備

Amazon GameLift の FlexMatch 機能を使用してプレイヤーのマッチメーキング機能をゲームに追加します。FlexMatch は、カスタムゲーム サーバーおよびリアルタイムサーバーのマネージド Amazon GameLift ソリューションで使用できます。

FlexMatchでは、マッチメーキングサービスとカスタマイズ可能なルールエンジンをペアリングします。これにより、ゲームに適切なプレイヤー属性とゲームモードに基づいてプレイヤーどおしをマッチングする方法を設計できます。また、FlexMatch でプレイヤーグループを編成してゲーム内に配置するための仕組みを管理できます。カスタムのマッチメーキングの詳細については、「FlexMatch ルールセットの例」を参照してください。

さらに、FlexMatch ではキューの特徴を構築できます。試合が作成されると、試合の詳細が FlexMatch から選択したキューに渡されます。キューは、Amazon GameLift フリートで利用可能なホストリソースを検索し、その試合の新しいゲームセッションを開始します。

このセクションのトピックでは、ゲームサーバーとゲームクライアントにマッチメーキングサポートを追加する方法について説明します。ゲームのマッチメーカーを作成する方法については、「Amazon GameLift FlexMatch マッチメーカーの構築」を参照してください。FlexMatchの動作に関する追加情報は、「Amazon GameLift FlexMatch の仕組み」 を参照してください。

## ゲーム クライアントへの FlexMatch の追加

このトピックでは、クライアント側ゲームサービスに FlexMatch マッチメーキングのSupportを追加する方法について説明しています。FlexMatch を Amazon GameLift マネージド ホスティング、または別のホスティングソリューションで使用している場合でも、プロセスは基本的に同じです。FlexMatch とゲームのカスタムマッチメーカーのセットアップ方法の詳細はこちらのトピックを参照してください。

- FlexMatch と Amazon GameLift ホスティングの統合
- Amazon GameLift FlexMatch の仕組み
- Amazon GameLift FlexMatch マッチメーカーの構築
- FlexMatch ルールセットの例

FlexMatch マッチメーキングをゲームで有効にするには、以下の機能を追加する必要があります。

- 1人以上のプレイヤー（必要な）のマッチメーキングのリクエストを準備します。
- マッチメーキングリクエスト（必要な）のステータスを追跡します。
- プレイヤーによる試合案（オプション）の承諾をリクエストします。
- ゲームセッションが新しい試合に対して作成されたら、プレイヤー接続情報を取得してゲームに参加します。

## プレイヤーによるマッチメーキングのリクエスト準備

ゲームクライアントがクライアント側のゲームサービスを通してマッチメーキングリクエストを作成することを強くお勧めします。信頼されたソースを使用することで、ハッキングの試みや偽のプレイヤーデータからより簡単に保護できます。ゲームがセッションディレクトリサービスを使用している場合、これはマッチメーキングリクエストを処理するための優れたオプションです。

クライアントサービスを準備するには、次のタスクを実行します。

- Amazon GameLift API を追加します。クライアントサービスは　AWS SDK のパートである Amazon GameLift API の機能を使用します。AWS SDKの詳細と最新バージョンのダウンロード方法についての詳細はこちら、「クライアントサービス用の Amazon GameLift SDK」を参照してください。ゲームクライアントサービスプロジェクトにこの SDK を追加します。
- [Set up a matchmaking ticket system]（マッチメーキングチケットシステムのセットアップ）すべてのマッチメーキングリクエストに、一意のチケット ID を割り当てる必要があります。一意の ID を生成し、それを新しいマッチングリクエストに割り当てるメカニズムが必要です。チケット ID には、最大 128 文字の文字列形式を使用できます。
- [Get matchmaker information]（マッチメーカー情報を取得する）使用する予定のマッチメーキング設定の名前を取得します。また、マッチメーカーの必須プレイヤー属性のリストも必要です。この属性は、マッチメーカーのルールセットで定義されています。
- [Get player data]（プレイヤーデータを取得します）各プレイヤーの関連データを取得する方法を設定します。たとえば、プレイヤーがゲームにスロットされる可能性が高いリージョンごとのプレイヤー ID、プレイヤー属性値、および更新されたレイテンシーデータなどがあります。
- (オプション) マッチバックフィルを有効にする。既存のマッチングされたゲームをバックフィルする方法を決定します。マッチメーカーのバックフィルモードが [手動] に設定されている場合、バックフィルサポートをゲームに追加できます。バックフィルモードが [自動] に設定されている場合、個々のゲームセッションに対して設定をオフにする方法が必要になる可能性があります。マッチングバックフィルの管理の詳細については、「FlexMatch を使用して既存のゲームをバックフィル」を参照してください。

# プレイヤーのマッチメーキングのリクエスト

クライアントサービスにコードを追加し、 FlexMatch マッチメーカーへのマッチメーキングリクエストを作成して管理します。FlexMatch マッチメーキングをリクエストするプロセスは、Amazon GameLiftManaged ホスティングで FlexMatch を使用するゲームと、スタンドアロンソリューションとして FlexMatch を使用するゲームでも同じです。

マッチメーキングリクエストを作成します。

- Amazon GameLift API StartMatchmaking の呼び出し 各リクエストには、以下の情報が必要です。

  マッチメーカー

  リクエストに使用するマッチメーキング設定の名前を指定します。FlexMatch は各リクエストを指定されたマッチメーカーのプールに配置し、リクエストはマッチメーカーが設定されている方法に基づいて処理されます。これには、プレイヤーにマッチングの承諾をリクエストするかどうか、生成されたゲームセッションを配置する際にどのようなキューを使用するかなど、強制的な時間制限が含まれます。マッチメーカーとルール設定の詳細については、「 FlexMatch マッチメーカーの設計」を参照してください。

  チケット ID

  リクエストに割り当てられている一意のチケット ID。イベントや通知などリクエストに関連するものすべてが、チケット ID を参照します。

  プレイヤーデータ

  マッチングを作成する対象のプレイヤーのリスト。リクエスト内のプレイヤーのいずれかが、マッチルールと最小レイテンシーに基づいてマッチ要件を満たしていない場合、マッチメーキングリクエストが成功することはありません。マッチリクエストには最大 10 人のプレイヤーを含めることができます。リクエストに複数のプレイヤーがいる場合、 FlexMatch は 1 つの試合を作成し、すべてのプレイヤーを同じチーム (ランダムに選択) に割り当てます。リクエストに含まれるプレイヤーが多すぎて、マッチチームの 1 つに収まらない場合、リクエストは一致しません。たとえば、2v2 のマッチ (2 人のプレイヤーで構成される 2 つのチーム) を作成するようにマッチメーカーを設定した場合は、3 人以上のプレイヤーを含むマッチメーキングリクエストを送信することはできません。

> **Note**
>
> プレイヤー (プレーヤー ID によって識別) は、一度に 1 つのアクティブなマッチメーキングリクエストにのみ含めることができます。プレイヤーの新しいリクエストを作成すると、同じプレイヤー ID のアクティブなマッチメーキングチケットは自動的にキャンセルされます。

リストされたプレイヤーごとに、次のデータを含めます。

- [Player ID ] ( プレイヤー ID ) 各プレイヤーごとにユニークなプレイヤー ID の生成が必要です。[Generate player IDs] ( プレイヤー ID の生成 ) を参照してください。
- [Player attributes] ( プレイヤー属性 ) 使用されているマッチメーカーがプレイヤー属性を呼び出した場合、リクエストは各プレイヤーにそれらの属性を提供する必要があります。必須のプレイヤー属性は、マッチメーカーのルールセット内で定義され、属性のデータ型も指定されます。プレイヤー属性は、ルールセットで属性のデフォルト値が指定された場合のみ、オプションとなります。マッチングリクエストが必要なプレイヤー属性をすべてのプレイヤーに提供しない場合、マッチメーキングリクエストは成功しません。マッチメーカーのルールセットおよびプレイヤー属性の詳細については、「FlexMatch ルールセットの構築」および「FlexMatch ルールセットの例」を参照してください。
- [Player latencies] ( プレイヤーレイテンシー ) 使用されているマッチメーカーにプレイヤーレイテンシールールがある場合、リクエストは各プレイヤーのレイテンシーを報告する必要があります。プレイヤーレイテンシーデータは、プレイヤーごとに 1 つ以上の値が表示されたリストです。これは、マッチメーカーのキューのリージョンで、プレイヤーに発生するレイテンシーを表しています。プレイヤーのレイテンシー値がリクエストに含まれていない場合、プレイヤーがマッチングされることはなく、リクエストは失敗します。

マッチングリクエストの詳細を取得する。

- マッチングリクエストが送信されたら、[DescribeMatchmaking] ( マッチメーキングの説明 ) をリクエストのチケット ID とともに呼び出すことによって、リクエストの詳細を表示できます。この呼び出しは、現在のステータスを含むリクエスト情報を返します。リクエストが正常に完了すると、チケットにはゲームクライアントがマッチングに接続するために必要な情報も含まれます。

マッチングリクエストをキャンセルする。

- マッチングリクエストは、[StopMatchmaking]（マッチメーキングの停止）をリクエストのチケット ID とともに呼び出すことによって、いつでもキャンセルできます。

## マッチメーキングイベントの追跡

マッチメーキングプロセスに関して Amazon GameLift がマッチメーキングプロセスのために発行するイベントを追跡するために通知を設定します。SNS トピックを作成するか、 Amazon EventBridge を使用することで、通知を直接設定できます。通知の設定の詳細については、「FlexMatch イベント通知をセットアップする」を参照してください。通知を設定したら、必要に応じてイベントと応答を検出するために、クライアントサービスでリスナーを追加します。

また、通知なしでかなりの期間が経過した場合に、ステータスの更新を定期的にポーリングして、通知をバックアップすることをお勧めします。マッチメーキングのパフォーマンスへの影響を最小限に抑えるには、マッチメーキングチケットが送信されてから 30 秒以上待ってからか、最後に受信した通知の後にのみポーリングしてください。

[DescribeMatchmaking]（マッチメーキングの説明）をリクエストのチケット ID とともに呼び出して、現在のステータスを含むマッチメーキングリクエストチケットを取得します。最大で 10 秒に 1 回ポーリングすることをお勧めします。この方法は、使用量が少ない開発シナリオでのみ使用します。

> **Note**
>
> 本稼働前の負荷テストなど、大量のマッチメーキングを使用する前に、イベント通知を使用してゲームをセットアップする必要があります。パブリックリリースのすべてのゲームでは、ボリュームに関係なく通知を使用する必要があります。継続的なポーリング方法は、マッチメーキングの使用量が少ない開発中のゲームにのみ適しています。

## プレイヤーの承諾をリクエスト

プレイヤーの承諾が有効になっているマッチメーカーを使用している場合、クライアントサービスにコードを追加し、プレイヤーの承諾プロセスを管理します。プレイヤーの承諾をマネージドするプロセスは、Amazon GameLift マネージドホスティングで FlexMatch を使用するゲームと、スタンドアロンソリューションとして FlexMatch を使用するゲームでも同じです。

マッチング案のプレイヤー承諾をリクエストする。

1. [Detect when a proposed match needs player acceptance] ( マッチング案がプレイヤーの承諾を必要とするときを検出する ) マッチメーキングチケットをモニタリングし、ステータスの REQUIRES_ACCEPTANCE への変更を検出します。このステータスに変わったときに FlexMatch イベントがトリガーされます。MatchmakingRequiresAcceptance。
2. [Get acceptances from all players] ( すべてのプレイヤーから承諾を得る ) 提案されたマッチング詳細を、マッチメーキングチケットで各プレイヤーに提示するメカニズムを作成します。プレイヤーは、マッチング案の承諾または拒否を示すことができる必要があります。マッチングの詳細は、DescribeMatchmaking ( マッチメーキングの説明 ) を呼び出して取得できます。マッチメーカーがマッチング案を取り消して続行する前に、プレイヤーには返答のために限られた時間が与えられます。
3. [Report player responses to FlexMatch.] ( プレイヤーのレスポンスを FlexMatch に報告する。 ) AcceptMatch を呼び出し、プレイヤーの承諾または拒否の応答を報告します。マッチメーキングリクエストのすべてのプレイヤーが承諾した場合にのみマッチングが進められます。
4. 承諾されなかったチケットを処理する。マッチング案でプレイヤーがマッチングを拒否するか、承諾の制限時間内に応答しなかった場合、リクエストは失敗します。試合を承諾したプレイヤーのチケットは、自動的にチケットプールに戻されます。試合を承諾しなかったプレイヤーのチケットは違反ステータスになり、処理が中断されます。複数のプレイヤーがいるチケットの場合、チケット内のいずれかのプレイヤーが試合を受け入れなかった場合、チケット全体が違反します。

## 試合にConnectする

正常に完了したマッチング (ステータス COMPLETED またはイベント MatchmakingSucceeded) を処理するために、クライアントサービスにコードを追加します。これにはマッチングのプレイヤーへの通知と、ゲームクライアントへ接続情報の提供が含まれます。

Amazon GameLift マネージドホスティングを使用するゲームでは、マッチメーキングリクエストが正常に実行されると、ゲームセッション接続情報がマッチメーキングチケットに追加されます。DescribeMatchmaking を呼び出して、完了したマッチメーキングチケットを取得します。接続情報には、ゲームセッションの IP アドレスとポート、および各プレイヤー ID に対するプレイヤーセッション ID が含まれます。詳細については、「GameSessionConnectionInfo」を参照してください。ゲームクライアントはこの情報を使用して、試合をホストしているゲームセッションに直接Connectします。接続リクエストには、プレイヤーセッション ID とプレイヤー ID が含まれている必

要があります。このデータは接続されたプレイヤーを、チームの割り当てを含むゲームセッションのマッチングデータに関連付けます (「GameSession」を参照)。

Amazon GameLift FleetIQ など、他のホスティングソリューションを使用するゲームでは、試合のプレイヤーが適切なゲームセッションに接続を有効にできるようメカニズムを構築する必要があります。

## マッチメーキングリクエストのサンプル

これらのコードスニペットは、さまざまなマッチメーカーに対するマッチメーキングリクエストを構築します。説明されているように、リクエストではマッチメーカーのルールセットの定義に従い、使用されているマッチメーカーで必要なプレイヤー属性を提供する必要があります。提供された属性では、ルールセット内で定義されている同じデータ型、数値 (N)、または文字列 (S) を使用する必要があります。

```
# 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)
```

# Amazon GameLift ホストゲームサーバーへの FlexMatch の追加

このトピックでは、Amazon GameLift マネージドホスティングを使用しているカスタムゲームサーバーに FlexMatch マッチメーキングサポートを追加する方法について説明します。FlexMatch をゲームに追加する方法の詳細については、以下のトピックを参照してください。

- Amazon GameLift FlexMatch の仕組み
- FlexMatch と Amazon GameLift ホスティングの統合

このトピックの情報では、「ゲームサーバーへの Amazon GameLift の追加」に示すように、Amazon GameLift Server SDK をゲームサーバープロジェクトに正常に統合したことを前提としています。この作業を完了すると、必要なほとんどのメカニズムが整います。このトピックの各セクションでは、FlexMatch でセットアップされたゲームを処理するために必要な残りの作業について説明します。

## マッチメーキング用のゲームサーバーの設定

マッチングされるゲームを処理するようにゲームサーバーを設定するには、次のタスクを完了します。

1. マッチメーキングで作成されたゲームセッションを開始します。新しいゲームセッションをリクエストするために、Amazon GameLift は `onStartGameSession()` リクエストをゲームセッションオブジェクトと共にゲームサーバーに送信します (「ゲームセッション」を参照)。ゲームサーバーは、カスタマイズされたゲームデータを含むゲームセッション情報を使用して、リクエストされたゲームセッションを開始します。詳細については、「ゲームセッションを開始する」を参照してください。

   マッチングされたゲームの場合、ゲームセッションオブジェクトには一連のマッチメーカーデータも含まれます。マッチメーカーデータには、マッチング用の新しいゲームセッションを開始するためにゲームサーバーで必要な情報が含まれます。これには、マッチングのチーム構造、チームの割り当て、およびゲームに関連する可能性がある特定のプレイヤー属性が含まれます。たとえば、ゲームで平均プレイヤースキルを基準に特定の機能またはレベルのロックを解除したり、プレイヤーの設定に基づいてマップを選択したりする場合があります。詳細については、「マッチメーカーデータの操作」を参照してください。

2. プレイヤーの接続を処理します。ゲームクライアントは、マッチングされたゲームに接続するときに、プレイヤー ID とプレイヤーセッション ID を参照します (「新しいプレイヤーの評価」を参照してください)。ゲームサーバーは、受信プレイヤーをマッチメーキングデータのプレイヤー情報に関連付けるためにプレイヤー ID を使用します。マッチメーカーデータは、プレイヤーのチームの割り当てを識別し、ゲームのプレイヤーを正しく表すためにその他の情報を提供します。

3. プレイヤーがいつゲームから離れたかをレポートします。ゲームサーバーがサーバー API `RemovePlayerSession()` を呼び出して、削除されたプレイヤーをレポートすることを確認します (「セッションの終了したプレイヤーの報告」を参照)。既存のゲームの空きスロットを満たすために FlexMatch バックフィルを使用している場合、このステップは重要です。ゲームがクライアント側のゲームサービスでバックフィルリクエストを開始する場合、これは非常に重要

です。バックフィルの実行の詳細については、「FlexMatch を使用して既存のゲームをバックフィル」を参照してください。

4. マッチングされる既存のゲームセッション用の新しいプレイヤーをリクエストします (オプション)。既存のマッチングされたゲームをバックフィルする方法を決定します。マッチメーカーのバックフィルモードが [手動] に設定されている場合、バックフィルサポートをゲームに追加できます。バックフィルモードが [自動] に設定されている場合、個々のゲームセッションに対して設定をオフにする方法が必要になる可能性があります。たとえば、ゲームで特定ポイントに達した時点でゲームセッションのバックフィリングを停止したい場合があります。マッチングバックフィルの管理の詳細については、「FlexMatch を使用して既存のゲームをバックフィル」を参照してください。

## マッチメーカーデータの操作

ゲームサーバーは、GameSession オブジェクトでゲーム情報を認識して使用できる必要があります。Amazon GameLift サービスは、ゲームセッションが開始または更新されるたびに、これらのオブジェクトをゲームサーバーに渡します。コアゲームセッション情報には、ゲームセッション ID と名前、最大プレイヤー数、接続情報、およびカスタムゲームデータ (ある場合) が含まれます。

FlexMatch を使用して作成されたゲームセッションの場合、 `GameSession` オブジェクトには一連のマッチメーカーデータも含まれます。このオブジェクトは、一意の ID に加えて、マッチングを作成したマッチメーカーを識別し、チーム、チーム割り当て、およびプレイヤーについて記述します。元のマッチメーキングリクエストからのプレイヤー属性も含まれます (Player オブジェクトを参照)。これにはプレイヤーのレイテンシーは含まれません。マッチバックフィルなどの現在のプレイヤーでレイテンシーデータが必要な場合は、更新データを取得することをお勧めします。

> **Note**
>
> マッチメーカーデータは、完全なマッチメーキング設定 ARN を指定します。これは、設定名、AWS アカウント、リージョンを特定します。マッチバックフィルをゲームクライアントまたはサービスからリクエストする場合は、設定名のみが必要です。":matchmakingconfiguration/" の後の文字列を解析すると、設定名を抽出できます。ここに示す例では、マッチメーキング設定の名前は「MyMatchmakerConfig」です。

次の JSON は、マッチメーキングデータの一般的なセットを示しています。この例では、2 つのプレイヤーゲームを示します。プレイヤーはスキル評価および達成した最高レベルに基づいてマッチングされます。マッチメーカーでは、キャラクターに基づいてもマッチングされ、マッチングされたプ

レイヤーは少なくとも 1 つの共通マップ設定を持ちます。このシナリオでは、ゲームサーバーは最優先されるマップを決定し、ゲームセッション内でそのマップを使用できます。

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

# FlexMatch を使用して既存のゲームをバックフィル

マッチバックフィルは、既存のマッチ済みのゲームセッションの新しいプレイヤーを見つけるために,FlexMatch メカニズムを使用します。プレイヤーは任意のゲームにいつでも追加できますが [Join a player to a game session] ( ゲームセッションへのプレイヤーの参加 ) を参照)、マッチバックフィルにより、新しいプレイヤーが既存のプレイヤーと同じ条件を満たすことができます。また、マッチバックフィルは新しいプレイヤーをチームに割り当て、プレイヤーの受け入れを管理し、更新されたマッチ情報をゲームサーバーに送信します。マッチバックフィルの詳細については「FlexMatch マッチメーキングプロセス」を参照してください。

> **Note**
>
> 現在、FlexMatch バックフィルは、Realtimeサーバーを使用しているゲームで使用することはできません。

バックフィルメカニズムには、次の 2 種類があります。

- 最大許容プレイヤー数よりも少ないプレイヤーで開始したゲームセッションをフィルするには、自動バックフィルを有効にします。
- 進行中のゲームセッションからドロップアウトしたプレーヤーを置き換えるには、バックフィルリクエストを送信する機能をゲームサーバーに追加します。

## 自動バックフィルの起動

自動マッチバックフィルでは、フィルされていない単独または複数のプレーヤースロットでゲームセッションが開始されるたびに、Amazon GameLift が自動的にバックフィルリクエストをトリガーします。この機能により、マッチしたプレイヤーの最小数が見つかるとすぐにゲームを開始し、残りのスロットを後で埋めることができるようになります。自動バックフィルはいつでも停止できます。

例として、6 ー 10 人のプレーヤーを保持できるゲームを考えてみましょう。FlexMatch は 最初6 人のプレイヤーを検索し、マッチングを行い、新しいゲームセッションを開始します。自動バックフィルを使用すると、新しいゲームセッションはすぐに 4 人の追加プレーヤーを要求することができます。ゲームの性質によっては、ゲームセッション中にいつでも新しいプレーヤーが参加できるように許可したいかもしれません。または、初期セットアップフェーズとゲームプレイ開始前に自動バックフィルを停止することもできます。

自動バックフィルをゲームに追加するには、ゲームに以下の変更を加えます。

1. [Enable automatic backfill] ( 自動バックアップを有効にする。 ) 自動バックフィルはマッチメーキング設定で管理します。有効にすると、そのマッチメーカーで作成されたすべてのマッチされたゲームセッションで使用されます。Amazon GameLift は、ゲームセッションがゲームサーバーで開始され次第、完全ではないゲームセッションへのバックフィルリクエストの生成を開始します。

   自動バックフィルを有効にするには、マッチング設定を開き、バックフィルモードを [自動] に設定します。詳細については、「マッチメーキング設定の作成」を参照してください。

2. [Turn on backfill prioritization] ( バックフィル優先設定の起動 ) マッチメーキングプロセスをカスタマイズして、新しいマッチを作成する前に、バックフィルリクエストのフィリングを優先させます。マッチメーキングルールセットで、アルゴリズムコンポーネントを追加し、バックフィル優先度を「高」に設定します。詳細については、「マッチアルゴリズムのカスタマイズ」を参照してください。
3. 新しいマッチングデータで既存のゲームセッションを更新します。Amazon GameLift は Server SDK コールバック関数を使用して、マッチング情報でゲームサーバーを更新しますonUpdateGameSession[ Initialize the server process] ( サーバープロセスを初期化する ) を参照してください)。バックフィルアクティビティの結果として更新されたゲームセッションオブジェクトを処理するために、コードをゲームサーバーに追加します。詳細については、「ゲームサーバー上のマッチデータの更新」を参照してください。
4. ゲームセッションの自動バックフィルを無効にする。個々のゲームセッション中はいつでも自動バックフィルの停止を選択できます。自動バックフィルを停止するには、Amazon GameLift API コール StopMatchmaking を実行するコードをゲームクライアントまたはゲームサーバーに追加します。この呼び出しにはチケット ID が必要です。最新のバックフィルリクエストのバックフィルチケット ID を使用します。この情報は、前のステップで説明されているように、ゲームセッションのマッチメーキングデータから取得できます。

## バックフィルリクエストの送信 (ゲームサーバーから)

ゲームセッションをホストするゲームサーバーのプロセスから直接、マッチバックフィルリクエストを開始できます。サーバープロセスは、現在ゲームに接続しているプレイヤーのおよび空のプレイヤースロットのステータスについての最新の情報を持ちます。

このトピックは、既に必要なFlexMatchコンポーネントを構築済みであり、マッチメーキングプロセスをゲームサーバーとクライアント側のゲームサービスに正常に追加済みであることを前提としています。FlexMatch のセットアップの詳細については、「FlexMatch と Amazon GameLift ホスティングの統合」を参照してください。

ゲームのマッチバックフィルを有効にするには、以下の機能を追加する必要があります。

- マッチメーキングバックフィルリクエストをマッチメーカーに送信し、リクエストのステータスを追跡する。
- ゲームセッションのマッチ情報を更新する。「ゲームサーバー上のマッチデータの更新」を参照してください。

他のサーバー機能と同様に、ゲームサーバーはAmazon GameLift Server SDK を使用します。この SDK は C++ および C# で使用できます。

ゲームサーバーからマッチバックフィルリクエストを作成するには、次のタスクを完了します。

1. [Trigger a match backfill request] ( マッチバックフィルリクエストをトリガーします ) 一般的に、マッチされたゲームに 1 つ以上の空きプレイヤースロットがある場合はいつでも、バックフィルリクエストを開始できます。バックフィルリクエストを、重要なキャラクターの役割を埋めるためや、チームのバランスを取るためなどの特定の状況に結びつけることもできます。また、ゲームセッションの継続時間に基づいてバックフィルアクティビティを制限することもできます。
2. [Create a backfill request] ( バックフィルリクエストを作成します ) マッチバックフィルリクエストを作成して FlexMatch マッチメーカーに送信するためのコードを追加します。バックフィルリクエストは、これらのサーバー API を使用して処理されます。

   - [StartMatchBackfill] ( マッチバックフィルの開始 )
   - [StopMatchBackfill] ( マッチバックフィルの停止 )

   バックフィルリクエストを作成するには、次の情報を指定して `StartMatchBackfill` を呼び出します。バックフィルリクエストをキャンセルするには、バックフィルリクエストのチケット ID を指定して `StopMatchBackfill` を呼び出します。

   - ][Ticket ID ( チケットID ) マッチメーキングチケット ID を供給します (または自動生成させることもできます)。同じメカニズムを使用して、マッチメーキングリクエストおよびバックフィルリクエストにもチケット ID を割り当てます。マッチメーキングおよびバックフィルのチケットも同じ方法で処理されます。
   - [Matchmaker] ( マッチメイカー ) バックフィルリクエストに使用するマッチメーカーを特定します。一般的に、元のマッチの作成に使用したものと同じマッチメーカーを使用します。このリクエストはマッチメーキング設定 ARN を取ります。この情報は、ゲームセッションをアクティブ化した際に Amazon GameLift によってサーバープロセスに提供されたゲームセッションオブジェクト (GameSession) に保存されます。マッチメーキング設定 ARN は `MatchmakerData` プロパティに含まれています。
   - [Game session ARN] ( Game session ARN ) バックフィルされるゲームセッションを特定します。ゲームセッション ARN はサーバー API GetGameSessionId() を呼び出して取得できます。マッチメーキングプロセス中は、新しいリクエストのチケットにはゲームセッション ID がありません。一方、バックフィルリクエストのチケットにはあります。ゲームセッション

ID があるかどうかが、新しいマッチのチケットとバックフィルのチケットを見分ける方法の 1 つです。

- [Player data] ( プレイヤーデータ ) バックフィルするゲームセッション内のすべての現在のプレイヤーのプレイヤー情報 (Player)が含まれています。この情報により、マッチメーカーは現在ゲームセッション内にいるプレイヤーに対して最良のプレイヤーマッチを見つけることができます。各プレーヤーのチームメンバーシップを含める必要があります。バックフィルを使用していない場合は、チームを指定しないでください。ゲームサーバーがプレイヤー接続情報を正確にレポートしているなら、次のようにしてこのデータを取得できます。

  1. ゲームセッションをホストしているサーバープロセスには、現在どのプレイヤーがゲームセッションに接続しているかについての最新情報があります。
  2. プレイヤ ID、属性、およびチームの割り当てを取得するには、ゲームセッションオブジェクト (GameSession ( ゲームセッション ) の `MatchmakerData` プロパティからプレイヤーデータを抽出します (「マッチメーカーデータの操作」を参照)。マッチメーカーデータには、ゲームセッションにマッチされたことのあるすべてのプレイヤーが含まれています。そのため、現在接続しているプレイヤーのみのプレイヤーデータをプルする必要があります。
  3. プレイヤーレイテンシーについては、マッチメーカーがレイテンシーデータを呼び出す場合、新しいレイテンシー値をすべての現在のプレイヤーから収集し、それを各 `Player` オブジェクトに含めます。レイテンシールールが省略されマッチメーカーにレイテンシールールがある場合、リクエストは正常にマッチされません。バックフィルリクエストには、ゲームセッションが現在置かれているリージョンのレイテンシーデータのみが必要です。ゲームセッションのリージョンは `GameSession` オブジェクトの `GameSessionId` プロパティから取得できます。この値はリージョンを含む ARN です。

3. [Track the status of a backfill request] ( バックフィルリクエストのステータスを追跡する ) Amazon GameLift は Server SDK コールバック関数 を使用して、バックフィルリクエストのステータスに関してゲームサーバーを更新しますonUpdateGameSession (「 サーバープロセスを初期化する」を参照)。ステータスメッセージ (およびバックフィルリクエストゲームサーバー上のマッチデータの更新 が成功した結果として更新されたゲーム セッションオブジェクト) を処理するコードを追加します。

   マッチメーカーで処理できるゲームセッションからのマッチバックフィルリクエストは一度に 1 つだけです。リクエストをキャンセルする必要がある場合は、[StopMatchmaking] ( マッチメーキングの停止 ) を呼び出します。リクエストを変更する必要がある場合は、`StopMatchBackfill` を呼び出してから、更新されたリクエストを送信します。

# バックフィルリクエストの送信 (クライアントサービスから)

バックフィルリクエストをゲームサーバーから送信する代わりに、クライアント側のゲームサービスから送信できます。このオプションを使用するには、クライアント側のサービスがゲームセッションアクティビティとプレイヤー接続の現在のデータにアクセスできる必要があります。ゲームがセッションディレクトリサービスを使用している場合に適した選択です。

このトピックは、既に必要なFlexMatchコンポーネントを構築済みであり、マッチメーキングプロセスをゲームサーバーとクライアント側のゲームサービスに正常に追加済みであることを前提としています。FlexMatch のセットアップの詳細については、「FlexMatch と Amazon GameLift ホスティングの統合」を参照してください。

ゲームのマッチバックフィルを有効にするには、以下の機能を追加する必要があります。

- マッチメーキングバックフィルリクエストをマッチメーカーに送信し、リクエストのステータスを追跡する。
- ゲームセッションのマッチ情報を更新する。ゲームサーバー上のマッチデータの更新を参照してください

他のクライアント機能と同様に、クライアント側のゲームサービスは、AWS SDK と Amazon GameLift API を使用します。この SDK は、C++、C#、およびその他いくつかの言語で使用できます。クライアント API の一般的な説明については、Amazon GameLift サービス API リファレンスを参照してください。このリファレンスには、Amazon GameLift 関連アクションの低レベルサービス API についての説明があり、言語固有のリファレンスガイドへのリンクが含まれています。

クライアント側のゲームサービスを設定してマッチしたゲームをバックフィルするには、次のタスクを完了します。

1. [Trigger a request for backfilling.] ( バックフィルのリクエストをトリガーします ) 。一般的に、マッチされたゲームに 1 つ以上の空きプレイヤースロットがある場合はいつでも、ゲームによってバックフィルリクエストが開始されます。バックフィルリクエストを、重要なキャラクターの役割を埋めるためや、チームのバランスを取るためなどの特定の状況に結びつけることもできます。また、ゲームセッションの継続時間に基づいてバックフィルを制限することもできます。トリガーに使用したものが何であれ、少なくとも次の情報が必要になります。この情報は、ゲームセッション ID を指定して DescribeGameSessions を呼び出すことで、ゲームセッションオブジェクト (GameSession) から取得できます。

- [Number of currently empty player slots] ( 現在の空のプレーヤースロットの数 ) この値は、ゲームセッションの最大プレイヤー制限と現在のプレイヤー数から計算できます。現在のプレーヤー数は、ゲームサーバーが Amazon GameLift サービスと通信して、新しいプレイヤーの接続を検証したり、切断されたプレイヤーをレポートするたびに更新されます。

- [Creation policy] ( 作成ポリシー ) この設定は、ゲームセッションで現在新しいプレーヤーを受け入れているかどうかを示します。

ゲームセッションオブジェクトには他にも、ゲームセッション開始時間、カスタムゲームプロパティ、マッチメーカーデータなどの有用な情報が含まれています。

2. [Create a backfill request] ( バックフィルリクエストを作成します ) マッチバックフィルリクエストを作成して FlexMatch マッチメーカーに送信するためのコードを追加します。バックフィルリクエストは、これらのクライアント API を使用して処理されます。

   - StartMatchBackfill
   - StopMatchmaking

   バックフィルリクエストを作成するには、次の情報を指定して `StartMatchBackfill` を呼び出します。バックフィルリクエストはマッチメーキングリクエスト (「プレイヤーのマッチメーキングのリクエスト」を参照) に似ていますが、既存のゲームセッションも特定します。バックフィルリクエストをキャンセルするには、バックフィルリクエストのチケット ID を指定して `StopMatchmaking` を呼び出します。

   - ][Ticket ID ( チケットID ) マッチメーキングチケット ID を供給します (または自動生成させることもできます)。同じメカニズムを使用して、マッチメーキングリクエストおよびバックフィルリクエストにもチケット ID を割り当てます。マッチメーキングおよびバックフィルのチケットも同じ方法で処理されます。
   - [Matchmaker] ( マッチメイカー ) 使用するマッチメーキング設定の名前を特定します。一般的に、元のマッチの作成に使用したものと同じマッチメーカーをバックフィルに使用します。この情報は、マッチメーキング設定 ARN の下の、ゲームセッションオブジェクト (GameSession) の、`MatchmakerData` プロパティにあります。名前の値は「matchmakingconfiguration/」に続く文字列です。(たとえば、ARN 値「value "arn:aws:gamelift:us-west-2:111122223333:matchmakingconfiguration/MM-4v4」であれば、マッチメーキング設定名は「MM-4v4」です。)

- [Game session ARN] ( ゲームセッション ARN ) バックフィルされるゲ[ームセッションを指定します。ゲームセッションオブジェクトから GameSessionId プロパティを使用すると、この ID は必要な ARN 値を使用します。バックフィルリクエストのマッチメーキングチケット (MatchmakingTicket) には、処理中にゲームセッション ID がありますが、新しいマッチメーキングリクエストはマッチが成立するまでゲームセッション ID を取得しません。ゲームセッション ID があるかどうかが、新しいマッチのチケットとバックフィルのチケットを見分ける方法の 1 つです。
- [Player data] ( プレイヤーデータ ) バックフィルするゲームセッション内のすべての現在のプレイヤーのプレイヤー情報 (Player)が含まれています。この情報により、マッチメーカーは現在ゲームセッション内にいるプレイヤーに対して最良のプレイヤーマッチを見つけることができます。各プレーヤーのチームメンバーシップを含める必要があります。バックフィルを使用していない場合は、チームを指定しないでください。ゲームサーバーがプレイヤー接続情報を正確にレポートしているなら、次のようにしてこのデータを取得できます。

  1. ゲームセッション ID を指定して (DescribePlayerSessions) を呼び出し、ゲームセッションに現在接続しているすべてのプレイヤーを検出します。各プレーヤーセッションにはプレイヤー ID が含まれます。ステータスフィルタを追加してアクティブなプレーヤーのみを取得できます。
  2. ゲームセッションオブジェクト (GameSession) の MatchmakerData プロパティからプレイヤーデータをプルします (「マッチメーカーデータの操作」を参照)。前のステップで取得したプレイヤー ID を使用して、現在接続されているプレーヤーのみのデータを取得します。プレイヤーが切断された場合マッチメーカーのデータは更新されないため、現在のプレイヤーのみのデータを抽出する必要があります。
  3. プレイヤーレイテンシーについては、マッチメーカーがレイテンシーデータを呼び出す場合、新しいレイテンシー値をすべての現在のプレイヤーから収集し、それを Player オブジェクトに含めます。レイテンシールールが省略されマッチメーカーにレイテンシールールがある場合、リクエストは正常にマッチされません。バックフィルリクエストには、ゲームセッションが現在置かれているリージョンのレイテンシーデータのみが必要です。ゲームセッションのリージョンは GameSession オブジェクトの GameSessionId プロパティから取得できます。この値はリージョンを含む ARN です。

3. バックフィルリクエストのステータスを追跡します。マッチメーキングチケットのステータス更新をリッスンするコードを追加します。イベント通知 (推奨) またはポーリングを使用して、新しいマッチメーキングリクエストのチケットを追跡するセットアップのメカニズムを使用できます (マッチメーキングイベントの追跡 を参照)。バックフィルリクエストでプレイヤーの承認アクティビティをトリガーする必要はなく、プレイヤー情報はゲームサーバーで更新されますが、チケットのステータスを追跡してリクエストの失敗や再送信を処理する必要はあります。

マッチメーカーで処理できるゲームセッションからのマッチバックフィルリクエストは一度に 1 つだけです。リクエストをキャンセルする必要がある場合は、StopMatchmaking を呼び出します。リクエストを変更する必要がある場合は、StopMatchmaking を呼び出してから、更新されたリクエストを送信します。

マッチバックフィルリクエストが成功すると、ゲームサーバーは更新された GameSession オブジェクトを受信し、新しいプレイヤーがゲームセッションに参加するために必要なタスクを処理します。詳細については、「ゲームサーバー上のマッチデータの更新」を参照してください。

## ゲームサーバー上のマッチデータの更新

ゲーム内でマッチバックフィルリクエストを開始する方法にかかわらず、 Amazon GameLift がマッチバックフィルリクエストの結果として送信するゲームセッション更新をゲームサーバーで処理できる必要があります。

Amazon GameLift がバックフィルリクエストを(正常であってもそうでなくても)完了すると、コールバック関数onUpdateGameSessionを使用してゲームサーバーを呼び出します。この呼び出しには 3 つの入力パラメータが存在します。マッチバックフィルチケット ID、ステータスメッセージ、およびプレーヤー情報を含む最新のマッチメーキングデータを含む GameSession オブジェクトです。ゲームサーバー統合の一部として、ゲームサーバーに次のコードを追加する必要があります。

1. onUpdateGameSession 関数を実装します。この関数は次のステータスメッセージ (updateReason) を処理できる必要があります。
   - MATCHMAKING_DATA_UPDATED 新しいプレーヤーが正常にゲームセッションにマッチされました。GameSession オブジェクトには、既存のプレイヤーおよび新しくマッチされたプレイヤーのプレイヤーデータを含む、更新されたマッチメーカーデータが含まれます。
   - BACKFILL_FAILED マッチバックフィル試行が内部エラーのために失敗しました。GameSession オブジェクトは変更されません。
   - BACKFILL_TIMED_OUT マッチメーカーが制限時間内にバックフィルマッチを見つけることができませんでした。GameSession オブジェクトは変更されません。
   - BACKFILL_CANCELLED マッチバックフィルリクエストが、StopMatchmaking(クライアント)または StopMatchBackfill (サーバー)の呼び出しによりキャンセルされました。GameSession オブジェクトは変更されません。
2. バックフィルマッチが成功するには、更新されたマッチメーカーデータを使用して、新しいプレイヤーがゲームセッションに接続した際にこれを処理します。少なくとも、プレイヤーがゲーム

を開始するために必要な他のプレイヤーの属性とともに、新しいプレイヤーのチーム割り当てを使用する必要があります。

3. ゲームサーバーのサーバー SDK アクション [ProcessReady]（ プロセスレディ ）に対する呼び出しで、 `onUpdateGameSession` コールバックメソッド名をプロセスパラメータとして追加します。

# Amazon GameLift FlexMatch リファレンス

このセクションには、Amazon GameLift FlexMatch でマッチメーキングを行うためのリファレンスドキュメントが含まれます。

トピック

- Amazon GameLift FlexMatch API リファレンス (AWS SDK)
- FlexMatch ルール言語
- FlexMatch マッチメーキングイベント

# Amazon GameLift FlexMatch API リファレンス (AWS SDK)

このトピックでは、Amazon GameLift FlexMatch に関するAPI オペレーションのタスクベースのリストを提供します。Amazon GameLift FlexMatch サービス API は `aws.gamelift` 名前空間の AWS SDK 内にあります。AWS SDK をダウンロードするか、Amazon GameLift API リファレンスドキュメントを参照してください。

Amazon GameLift FlexMatch は、Amazon GameLift ホスティングソリューション ( カスタムゲームサーバーまたはリアルタイムサーバー用のマネージドホスティング、Amazon GameLift FleetIQ による Amazon EC2 でのホスティングを含む ) でホストされているゲームで使用するためのマッチメーキングサービスを提供し、同様に、ピアツーピア、オンプレミス、またはクラウドコンピューティングプリミティブなど他のホスティングシステムにも提供します。Amazon GameLift のホスティングオプションの詳細については、Amazon GameLift 開発者ガイドを参照してください。

## マッチメーキングのルールとプロセスを設定する

これらのオペレーションを呼び出して、FlexMatch マッチメーカーを作成し、ゲームのマッチメーキングプロセスを設定し、試合とチームを作成するためのカスタムルールのセットを定義します。

マッチメーキング設定

- マッチメーキング設定の作成 グループまたはプレイヤーを評価してプレイヤーグループを構築する手順を含んだマッチメーキング設定を作成します。ホスティングに Amazon GameLift を使用する場合は、試合の新しいゲームセッションの作成方法を指定します。
- DescribeMatchmakingConfigurations – Amazon GameLift リージョンが定義されているマッチメーキング設定を取得します。

- UpdateMatchmakingConfiguration (マッチメーキング設定の更新) - マッチメーキング設定の設定を変更します。キュー。
- DeleteMatchmakingConfiguration (マッチメーキング設定の削除) - リージョンからマッチメーキング設定を削除します。

マッチメーキングのルールセット

- CreateMatchmakingRuleSet (マッチメーキングルールセットの作成) - プレイヤーのマッチングを検索するときに使用するルールのセットを作成します。
- DescribeMatchmakingRuleSets – Amazon GameLift リージョンで定義されているマッチメーキングルールセットを取得します。
- ValidateMatchmakingRuleSet (マッチメーキングルールセットの評価) - 一連のマッチメーキングルールの構文を検証します。
- DeleteMatchmakingRuleSet (マッチメーキングルールセットの削除) - リージョンからマッチメーキングルールセットを削除します。

## 1人または複数のプレイヤーの試合をリクエストする

以下のオペレーションをゲームクライアントサービスから呼び出して、プレイヤーのマッチメーキングリクエストを管理します。

- StartMatchmaking (マッチメーキングの開始) - 一緒にプレイするプレイヤーまたはグループのマッチメーキングをリクエストします。
- DescribeMatchmaking (マッチメーキングの記述) - ステータスなど、マッチメーキングリクエストの詳細を取得します。
- AcceptMatch – プレイヤーの承諾を必要とするマッチングの場合は、プレイヤーがマッチング案を受け入れたときに Amazon GameLift に通知します。
- StopMatchmaking (マッチメーキングの停止) - マッチメーキングリクエストをキャンセルします。
- StartMatchBackfill (マッチングバックフィルの開始) - 既存のゲームセッションの空のスロットを埋めるために、追加のプレイヤーマッチングをリクエストします。

## 利用可能なプログラミング言語

Amazon GameLift をサポートする AWS SDK は以下の言語で利用可能です。開発環境のサポートの詳細については、各言語のドキュメントを参照してください。

- C++ (SDK ドキュメント) (Amazon GameLift)
- Java (SDK ドキュメント) (Amazon GameLift)
- .NET (SDK ドキュメント) (Amazon GameLift)
- Go (SDK ドキュメント) (Amazon GameLift)
- Python (SDK ドキュメント) (Amazon GameLift)
- Ruby (SDK ドキュメント) (Amazon GameLift)
- PHP (SDK ドキュメント) (Amazon GameLift)
- JavaScript/Node.js (SDK ドキュメント) (Amazon GameLift)

# FlexMatch ルール言語

このセクションのリファレンストピックでは、Amazon GameLift FlexMatch で使用するマッチメーキングルールを構築するために使用される構文とセマンティクスについて説明します。マッチメーキングルールとルールセットの作成に関する詳細なヘルプについては、「FlexMatch ルールセットの構築」を参照してください。

トピック

- FlexMatch ルールセットスキーマ
- FlexMatch ルールセットプロパティの定義
- FlexMatch ルールタイプ
- FlexMatch のプロパティ式

## FlexMatch ルールセットスキーマ

FlexMatch ルールセットは、小規模対戦と大規模対戦のルールに標準スキーマを使用します。各セクションの詳細な説明については、「FlexMatch ルールセットプロパティの定義」を参照してください。

### 小規模対戦用のルールセットスキーマ

次のスキーマは、最大 40 人のプレイヤーの試合を構築するために使用されるルールセットのすべての可能なプロパティと許可された値を記載しています。

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

```
}
```

## 大規模対戦用のルールセットスキーマ

次のスキーマは、40 人以上のプレイヤーの試合を構築するために使用されるルールセットのすべての可能なプロパティと許可された値を記載しています。ルールセットで定義されているすべてのチームの maxPlayers 値が 40 を超える場合、 FlexMatch は大規模対戦ガイドラインに従って、このルールセットを使用するすべてのリクエストを処理します。

```
{
    "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 ルールセットプロパティの定義

このセクションでは、ルールセットスキーマ内の各プロパティを定義します。ルールセットの作成方法については、「FlexMatch ルールセットの構築」を参照してください。

**`name`**

ルールセットの説明ラベル この値は Amazon GameLift MatchmakingRuleSet リソースに割り当てられた名前には関連付けられていません。この値は、完了した試合を記述するマッチメーキングデータに含まれますが、Amazon GameLift プロセスでは使用されません。

許可される値: 文字列

必須? No

**`ruleLanguageVersion`**

使用されている FlexMatch プロパティ式言語のバージョン。

許可される値: 1.0

必須? Yes

**`playerAttributes`**

マッチメーキングリクエストに含まれ、マッチメーキングプロセスで使用されるプレイヤーデータの集合。また、マッチメーキングプロセスでデータが使用されていない場合でも、ゲームサーバーに渡されるマッチメーキングデータにプレイヤーデータを含めるように属性を宣言することもできます。

必須? No

**name**

マッチメーカーが使用するプレイヤー属性のユニークな名前。この名前は、マッチメーキングリクエストで参照されるプレイヤー属性名と一致する必要があります。

許可される値: 文字列

必須? Yes

**type**

プレイヤー属性値のデータ型。

許可される値:「string」 (文字列) 、「number」 (数値) 、「string_list」 (文字列リスト) 、「string_number_map」 (文字列番号マップ)

必須? Yes

**default**

マッチングリクエストをプレイヤーに提供しない場合、使用するデフォルト値。

許可された値:プレイヤー属性に許可される任意の値。

必須? No

**algorithm**

マッチメーキングプロセスをカスタマイズするためのオプションの構成設定。

必須? No

**strategy**

試合を構築するときに使用するメソッド。このプロパティが設定されていない場合、デフォルトの動作は「exhaustiveSearch」 (網羅的検索) です。

許可される値:

- 「exhaustiveSearch」 (網羅的検索) — 標準的なマッチング方法。FlexMatch は、一連のカスタムマッチルールに基づいてプール内の他のチケットを評価することにより、バッチ内の最も古いチケットを中心に試合を形成します。この戦略は、40人以下のプレーヤーの試合に使用されます。この戦略を使用する場合、`batchingPreference` は「random」 (ランダム) または「sorted」 (ソート) のいずれかに設定する必要があります。

- 「balanced」 (バランス) — 大きなマッチをすばやく形成するように最適化される方法 この戦略は、41から200人のプレーヤーの試合にのみ使用されます。チケットプールをあらかじめソートし、潜在的な試合を構築し、チームにプレーヤーを割り当てて、指定されたプレーヤー属性を使用して試合の各チームのバランスをとることで試合を形成します。たとえば、この戦略は、試合中の全チームの平均スキルレベルを均等化するために使用できます。この戦略を使用する場合、`balancedAttribute`を設定する必要があり、`batchingPreference`は「largestPopulation」 (最大人数) または「fastestRegion」 (最速リージョン) のいずれかに設定する必要があります。ほとんどのカスタムルールタイプは、この戦略では認識されません。

必須? Yes

**`batchingPreference`**

試合構築のチケットをグループ化する前に使用する事前ソート方法 チケットプールを事前にソートすると、特定の特性に基づいてチケットがまとめてバッチ処理され、最終試合のプレイヤー間で均一性が高まる傾向があります。

許可される値:

- 「random」 (ランダム) — `strategy` =「exhaustiveSearch」 (網羅的検索) の場合のみ有効 事前ソートは行われません。プール内のチケットはランダムにバッチ処理されます。これは、網羅的検索戦略のデフォルトの動作です。
- 「sorted」 (ソート) — `strategy` =「exhaustiveSearch」 (網羅的検索)の場合のみ有効。チケットプールは、`sortbyAttributes` に記載されているプレーヤー属性に基づいて事前にソートされています。
- 「largestPopulation」 (最大人数) — `strategy` =「balanced」 (バランス) の場合のみ有効。チケットプールは、プレーヤーが許容可能なレイテンシー値を報告するリージョン別に事前にソートされています。これは、バランスのとれた戦略のデフォルトの動作です。
- 「fastestRegion」 (最速リージョン) — `strategy`=「balanced」 (バランス) の場合のみ有効。チケットプールは、プレーヤーが許容可能なレイテンシー値を報告するリージョン別に事前にソートされています。結果の試合の完了には時間がかかりますが、すべてのプレイヤーのレイテンシーは低くなる傾向があります。

必須? Yes

**`balancedAttribute`**

バランスの取れた戦略で大規模対戦を構築する際に使用するプレイヤー属性の名前。

許可される値: `playerAttributes` と `type` =「number」 (数字) で宣言された属性。

必須? はい、 `strategy` =「balaned」 (バランス) の場合。

**`sortByAttributes`**

バッチ処理前にチケットプールを事前ソートするときに使用するプレイヤー属性のリスト。このプロパティは、網羅的検索戦略で事前ソートする場合のみ使用されます。属性リストの順序によって、ソート順序が決まります。FlexMatch は、英数字と数値に標準のソート規則を使用します。

許可される値: `playerAttributes` で宣言された属性。

必須? はい、 `batchingPreference` =「sorted」 (ソート) の場合。

**`backfillPriority`**

バックフィルチケットを照合するための優先順位付け方法。このプロパティは、FlexMatch がバックフィルチケットをバッチで処理するタイミングを決定します。このプロパティは、網羅的検索戦略で事前ソートする場合のみ使用されます。このプロパティが設定されていない場合、デフォルトの動作は「normal」 (ノーマル) です。

許可される値:

- 「normal」 (ノーマル) — チケットのリクエストタイプ (バックフィルまたは新規対戦) は、試合を形成する際に考慮されません。
- 「high」 (ハイ) — チケットバッチはリクエストタイプ (および経過時間) でソートされ、FlexMatch は最初にバックフィルチケットを照合しようとします。
- 「low」 (ロー) — チケットバッチはリクエストタイプ (および経過時間) でソートされ、FlexMatch は最初にバックフィルではないチケットを照合しようとします。

必須? No

**`expansionAgeSelection`**

対戦ルール拡張の待機時間を計算する方法。拡張は、一定時間が経過しても試合が完了しなかった場合に、対戦要件を緩和するために使用されます。待機時間は、すでに部分的に満たされた試合にあるチケットの経過時間に基づいて計算されます。このプロパティが設定されていない場合、デフォルトの動作は「最新」です。

許可される値:

- 「newest」 (最新) — 拡張待ち時間は、部分的に完了した試合で最新の作成タイムスタンプを持つチケットに基づいて計算されます。新しいチケットが待機時間クロックを再開できるので、拡張はゆっくりトリガーされる傾向があります。

- 「oldest」 (最古) — 拡張待ち時間は、部分的に完了した試合で最古の作成タイムスタンプを持つチケットに基づいて計算されます。拡張はより迅速にトリガーされる傾向があります。

必須? No

**teams**

試合におけるチームの構成。各チームのチーム名とサイズ範囲を提供します。ルールセットでは、少なくとも 1 つのチームを定義する必要があります。

**name**

チームのユニークな名前。チーム名は、ルールおよび拡張で参照できます。試合が成功すると、マッチメーキングデータで選手はチーム名で割り当てられます。

許可される値: 文字列

必須? Yes

**maxPlayers**

チームに割り当てることができるプレイヤーの最大数。

許可される値: 数値

必須? Yes

**minPlayers**

対戦を実行する前に割り当てる必要があるプレイヤーの最小数。

許可される値: 数値

必須? Yes

**quantity**

試合で作成するこのタイプのチームの数。数量が1より大きいチームは、付加番号 (「赤_1」、「赤_2」など) で指定します。このプロパティが設定されていない場合、デフォルト値は、1 です。

許可される値: 数値

必須? No

**rules**

対戦するプレイヤーの評価方法を定義する一連のルールステートメントを作成します。

必須? No

**name**

ルールのユニークな名前。ルールセット内のすべてのルールにはユニークな名前が必要です。ルール名は、このルールに関連するアクティビティを追跡するイベントログとメトリクスでも参照されます。

許可される値: 文字列

必須? Yes

**description**

ルールに関してテキストで示された説明。この情報を使用して、ルールの目的を特定できます。マッチメーキングプロセスでは使用しません。

許可される値: 文字列

必須? No

**type**

ルールステートメントのタイプ。各ルールタイプには、設定する必要がある追加のプロパティがあります。各ルールタイプの構造と使用方法の詳細については、「FlexMatch ルールタイプ」を参照してください。

許可される値:

- 「absoluteSort」 (絶対的ソート) — 数値を持つ指定されたプレイヤー属性と、バッチ内の最も古いチケットとの比較の両方に基づいて、バッチ内のチケットを並べ替える明示的なソート方法を使用するソート。
- 「collection」 (集合体) — 集合体内のプレイヤー属性や、複数のプレイヤーの一連の値など、集合体内の値を評価します。
- 「comparison」 (比較) — 2 つの値を比較します。
- 「compound」(複合) — ルールセット内の他のルールを論理的に組み合わせて複合マッチメーキングルールを定義します。40 人以下のプレイヤーのマッチでのみサポートされます。

- 「distance」 (距離) — 数値間の距離を測定します。
- 「BatchDistance」(バッチ距離) — 属性値の差を測定し、それを使用してマッチリクエストをグループ化します。
- 「distanceSort」 (距離ソート) — 数値を持つ指定されたプレイヤー属性と、バッチ内の最も古いチケットとの比較状況に基づいて、バッチ内のチケットを並べ替える明示的なソート方法を使用するソート。
- 「latency」 (レイテンシー) — マッチメーキングリクエストについて報告されるリージョンのレイテンシーデータを評価します。

必須? Yes

**expansions**

試合を完了できない場合に、時間の経過とともに試合要件を緩和するためのルール。試合を見つけやすくするために、徐々に適用される一連のステップとして拡張を設定します。デフォルトでは、FlexMatch はマッチに追加された最新のチケットの経過時間に基づいて待機時間を計算します。algorithm プロパティ `expansionAgeSelection` を使用して、拡張待ち時間の計算方法を変更できます。

拡張待ち時間は絶対値であるため、各ステップの待機時間は前のステップより長くなければなりません。たとえば、一連の拡張を段階的にスケジュールするには、30 秒、40 秒、50 秒の待機時間を使用します。待機時間は、マッチメーキング設定で設定される、対戦リクエストで許可された最大時間を超えることはできません。

必須? No

**target**

緩和されるルールセット要素。チームサイズのプロパティやルールステートメントのプロパティは緩和できます。構文は 「<component name>[<rule/team name>] <property name>」です。たとえば、チームの最小サイズを変更するには、次のようにします。`teams[Red, Yellow].minPlayers`。「minSkill」という名前の比較ルールステートメントの最小スキル要件を変更するには、次の手順を実行します。`rules[minSkill].referenceValue`。

必須? Yes

**steps**

**waitTimeSeconds**

ターゲットルールセット要素の新しい値を適用する前に待機する時間 (秒)。

必須? Yes

**value**

ターゲットルールセット要素の新しい値。

# FlexMatch ルールタイプ

## バッチ距離ルール

```
batchDistance
```

バッチ距離ルールでは、2 つの属性値の差異を測定します。バッチ距離ルールタイプは、マッチ数の多いものと小さいものの両方に使用できます。バッチ距離ルールには 2 つのタイプがあります。

- 数値属性値を比較する。例えば、このタイプのバッチ距離ルールでは、マッチのすべてのプレイヤー相互間のレベル差が 2 レベル以内であることが必要になる場合があります。このタイプでは、すべてのチケットの batchAttribute 間の最大距離を定義します。
- 文字列属性値を比較する。例えば、このタイプのバッチ距離ルールでは、マッチに参加しているすべてのプレイヤーが同じゲームモードをリクエストする必要がある場合があります。このタイプでは、FlexMatch がバッチを作成するために使用する batchAttribute 値を定義します。

バッチ距離ルールのプロパティ

- **batchAttribute** – バッチの作成に使用されるプレイヤー属性値。
- **maxDistance** – マッチングを成功させるための距離の最大値または最小値。数値属性の比較に使用されます。
- **partyAggregation**-複数のプレイヤー (パーティー) でチケットを FlexMatch がどのように処理するかを決定する値。有効なオプションには、チケットのプレイヤーに対して最小 (min)、最大 (max) および平均 (avg) の値を含まれます。デフォルトは avg です。

Example

例

```
{
  "name":"SimilarSkillRatings",
```

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

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

## 比較ルール

```
comparison
```

比較ルールでは、他の値とプレイヤー属性値を比較します。比較ルールには 2 つのタイプがあります。

- 参考値と比較する。例えば、このタイプの比較ルールでは、マッチしたプレイヤーが一定以上のスキルレベルを持っていることが求められる場合があります。このタイプでは、プレイヤー属性、参照値、比較演算を指定します。
- プレイヤー間で比較する。例えば、このタイプの比較ルールでは、試合のすべてのプレイヤーが異なるキャラクターを使用することが要求される場合があります。このタイプでは、プレイヤー属性と equal (=) または not-equal (!=) 比較演算のいずれかを指定します。参照値を指定しないでください。

> **Note**
>
> バッチ距離ルールは、プレーヤーの属性を比較する場合に効率的です。マッチメーキングの待ち時間を短縮するには、可能であればバッチ距離ルールを使用してください。

### 比較ルールのプロパティ

- **`measurements`** – 比較するプレイヤーの属性値。
- **`referenceValue`** – マッチング候補の測定値と比較するための値。

- **operation** – 測定値を基準値と比較する方法を決定する値。有効なオペレーションは <、<=、=、!=、>、>= です。
- **partyAggregation** – 複数のプレイヤー (パーティー) でチケットを FlexMatch がどのように処理するかを決定する値。有効なオプションには、チケットのプレイヤーに対して最小 (min)、最大 (max) および平均 (avg) の値を含まれます。デフォルトは avg です。

## 距離ルール

```
distance
```

距離ルールでは、プレイヤーのスキルレベル間の距離など、2 つの数値間の差異を測定します。例えば、距離ルールでは、すべてのプレイヤーが 30 時間以上ゲームをプレイしていることが必要になる場合があります。

> **Note**
>
> バッチ距離ルールは、プレーヤーの属性を比較する場合に効率的です。マッチメーキングの待ち時間を短縮するには、可能であればバッチ距離ルールを使用してください。

距離ルールのプロパティ

- **measurements** – 距離を測定するプレイヤー属性値。これは数値付きの属性でなければなりません。
- **referenceValue** – マッチング候補に対して距離を測定する数値。
- **minDistance/maxDistance** – マッチングを成功させるための距離の最大値または最小値。
- **partyAggregation** – 複数のプレイヤー (パーティー) でチケットを FlexMatch がどのように処理するかを決定する値。有効なオプションには、チケットのプレイヤーに対して最小 (min)、最大 (max) および平均 (avg) の値を含まれます。デフォルトは avg です。

## コレクションルール

```
collection
```

コレクションルールは、プレイヤー属性値のグループをバッチ内の他のプレイヤーの属性値または参照値と比較します。コレクションには、複数のプレイヤーの属性値、文字列リストであるプレイヤー

属性、またはその両方が含まれます。例えば、コレクションルールでは、チームが選択したプレイヤーのキャラクターを確認する場合があります。そのルールでは、チームに特定のキャラクターを少なくとも 1 人配置することが義務付けられる場合があります。

収集ルールのプロパティ

- **measurements** – 比較するプレイヤー属性値の収集。属性値は文字列のリストである必要があります。
- **referenceValue** – マッチング候補で測定の比較に使用する値 (または値の収集)
- **operation** – 一連の測定値を比較する方法を決定する値。有効なオペレーションには以下が含まれます。
  - `intersection` – このオペレーションでは、すべてのプレイヤーのコレクションで同じ値の数を測定します。intersection オペレーションを使用するルールの例については、「例 4: 明示的な並べ替えを使用して最適なマッチングを見つける」を参照してください。
  - `contains` – 指定された参照値を含むプレイヤー属性のコレクションの数を測定します。contains オペレーションを使用するルールの例については、「例 3: チームレベル要件とレイテンシーの制限を設定する」を参照してください。
  - `reference_intersection_count` – このオペレーションは、プレーヤー属性コレクション内のアイテムのうち、参照値コレクション内のアイテムと一致するアイテムの数を測定します。このオペレーションを使用して、複数の異なるプレーヤー属性を比較できます。複数のプレイヤー属性コレクションを比較するルールの例については、「例 5: 複数のプレイヤー属性間の交差を見つける」を参照してください。
- **minCount/maxCount** – 対戦を成功させるためのカウントの最小値または最大値。
- **partyAggregation** – 複数のプレイヤー (パーティー) でチケットを FlexMatch がどのように処理するかを決定する値。この値では、`union` を使用してパーティー内のすべてのプレイヤーのプレイヤー属性を組み合わせることができます。または、`intersection` を使用してパーティーに共通するプレイヤー属性を使用することもできます。デフォルトは `union` です。

## 複合ルール

```
compound
```

複合ルールでは、論理的な記述を用いて 40 人以下のプレイヤーでマッチを行います。1 つのルールセットで複数の複合ルールを使用できます。複数の複合ルールを使用する場合、一致するにはすべての複合ルールが満たされている必要があります。

拡張ルールを使用して複合ルールを拡張することはできませんが、基本ルールやサポートルールを拡張することはできます。

複合ルールプロパティ

- **statement** – 個々のルールを組み合わせて複合ルールを形成するために使用されるロジック。このプロパティで指定するルールは、ルールセットで以前に定義されている必要があります。複合ルールでは batchDistance ルールを使用できません。

  このプロパティは以下の論理演算子をサポートします。

  - and – 指定された 2 つの引数が true の場合、式は true と評価されます。
  - or – 指定された 2 つの引数のいずれか true の場合、式は true と評価されます。
  - not – 式内の引数の結果を逆にします。
  - xor – 引数のうち 1 つだけが true の場合、式は true と評価されます。

Example 例

次の例では、選択したゲームモードに基づいてさまざまなスキルレベルのプレイヤーをマッチングします。

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

## レイテンシールール

```
latency
```

レイテンシールールは、プレイヤーのレイテンシーをロケーションごとに測定します。レイテンシールールでは、レイテンシーが最大値を超える場所はすべて無視されます。レイテンシールールが受け入れるには、プレイヤーのレイテンシー値が少なくとも 1 つのロケーションで最大値を下回っている必要があります。maxLatency プロパティを指定することで、マッチ数が多い場合にこのルールタイプを使用できます。

レイテンシールールのプロパティ

- **maxLatency** – ロケーションの最大許容レイテンシー値。レイテンシーが最大値を下回るロケーションがチケットにない場合、そのチケットはレイテンシールールに一致しません。
- **maxDistance** – 各チケットのレイテンシーと距離の参照値との間の最大値。
- **distanceReference** – チケットのレイテンシーを比較するためのレイテンシー値。チケットが距離の参照値から最大距離内にある場合、一致に成功します。有効なオプションには最小 (min) および平均 (avg) プレイヤーレイテンシー値が含まれます。
- **partyAggregation** – 複数のプレイヤー (パーティー) でチケットを FlexMatch がどのように処理するかを決定する値。有効なオプションには、チケットのプレイヤーに対して最小 (min)、最大 (max) および平均 (avg) の値を含まれます。デフォルトは avg です。

> **Note**
>
> キューは、レイテンシールールに一致しないリージョンにゲームセッションを配置できます。キューのレイテンシーポリシーについて詳しくは、「プレイヤーレイテンシーポリシーを作成する」を参照してください。

### 絶対ソートルール

```
absoluteSort
```

絶対ソートルールでは、最初に追加されたチケットと比較して、指定されたプレイヤー属性に基づいてマッチメーキングチケットのバッチを並べ替えます。

絶対並べ替えルールのプロパティ

- **sortDirection** – マッチメーキングチケットをソートする順序。有効なオペレーションは ascending および descending です。
- **sortAttribute** – チケットをソートする基準となるプレイヤー属性。
- **mapKey**— プレイヤー属性がマップの場合、その属性をソートするオプション。有効なオペレーションは以下のとおりです。
  - minValue – 値が最も低いキーが最初になります。
  - maxValue – 値が最も高いキーが最初になります。
- **partyAggregation** – 複数のプレイヤー (パーティー) でチケットを FlexMatch がどのように処理するかを決定する値。有効なオプションには、パーティ内のプレイヤーの最小プレイヤー属性

(min)、最大プレイヤー属性 (max)、全プレイヤー属性の平均 (avg) などがあります。デフォルトは avg です。

Example

例

以下のルール例では、プレイヤーをスキルレベルでソートし、パーティーのスキルレベルを平均しています。

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

## 距離ソートルール

```
distanceSort
```

距離ソートルールでは、最初に追加されたチケットと指定されたプレイヤー属性の距離に基づいてマッチメーキングチケットのバッチを並べ替えます。

距離並べ替えルールのプロパティ

- **sortDirection** – マッチメーキングチケットを並べ替える方向。有効なオペレーションは ascending および descending です。
- **sortAttribute** – チケットをソートする基準となるプレイヤー属性。
- **mapKey**— プレイヤー属性がマップの場合、その属性をソートするオプション。有効なオペレーションは以下のとおりです。
  - minValue – バッチに最初に追加されたチケットの場合、最低値のキーを見つけます。
  - maxValue – バッチに最初に追加されたチケットの場合、最高値のキーを見つけます。
- **partyAggregation** – 複数のプレイヤー (パーティー) でチケットを FlexMatch がどのように処理するかを決定する値。有効なオプションには、チケットのプレイヤーに対して最小 (min)、最大 (max) および平均 (avg) の値を含まれます。デフォルトは avg です。

# FlexMatch のプロパティ式

プロパティ式は、マッチメーキングに関連する特定のプロパティを定義するために使用できます。プロパティ値を定義するときに、計算とロジックを使用できます。通常、プロパティ式は 2 種類の形式のいずれかになります。

- 個別のプレイヤーデータ
- 個々のプレイヤーデータの計算されたコレクション。

## 一般的なマッチメーキングプロパティ式

プロパティ式は、プレーヤー、チーム、またはマッチの特定の値を識別します。次の部分的な式は、チームとプレーヤーを特定する方法を示しています。

| 目標 | 入力 | 意味 | 出力 |
| --- | --- | --- | --- |
| マッチングの特定のチームを識別するには: | `teams[red]` | Red チーム | Team |
| マッチングの一連の特定のチームを識別するには: | `teams[red,blue]` | レッドチームとブルーチーム | List<Team> |
| マッチングのすべてのチームを識別するには: | `teams[*]` | すべてのチーム | List<Team> |
| 特定のチームのプレイヤーを識別するには: | `team[red] .players` | Red チームのプレイヤー | List<Player> |
| マッチの一連の特定のチームのプレイヤーを識別するには: | `team[red, blue].players` | マッチングのプレイヤー (チーム別にグループ分け) | List<List<Player>> |

| 目標 | 入力 | 意味 | 出力 |
| --- | --- | --- | --- |
| マッチングのプレイヤーを識別するには: | `team[*].players` | マッチングのプレイヤー (チーム別にグループ分け) | List<List<Player>> |

## プロパティ式の例

次の表は、前の例に基づいて構築されたプロパティ式を示しています。

| 式 | 意味 | 結果のタイプ |
| --- | --- | --- |
| `teams[red].players[playerId]` | Red チームに属するすべてのプレイヤーのプレイヤー ID | List<string> |
| `teams[red].players.attributes[skill]` | Red チームに属するすべてのプレイヤーの「skill」属性 | List<number> |
| `teams[red,blue].players.attributes[skill]` | レッドチームとブルーチームの全プレイヤーの「スキル」属性 (チーム別) | List<List<number>> |
| `teams[*].players.attributes[skill]` | マッチングに属するすべてのプレイヤーの「skill」属性 (チーム別にグループ分け) | List<List<number>> |

## プロパティ式の集約

プロパティ式は、以下の関数や関数の組み合わせを使用してチームデータを集約するために使用できます。

| 集約 | 入力 | 意味 | 出力 |
| --- | --- | --- | --- |
| `min` | List<number> | リスト内のすべての数値の最小値を取得します。 | number |

| 集約 | 入力 | 意味 | 出力 |
| --- | --- | --- | --- |
| max | List<number> | リスト内のすべての数値の最大値を取得します。 | number |
| avg | List<number> | リスト内のすべての数値の平均値を取得します。 | number |
| median | List<number> | リスト内のすべての数値の中央値を取得します。 | number |
| sum | List<number> | リスト内のすべての数値の合計値を取得します。 | number |
| count | List<?> | リスト内の要素の数を取得します。 | number |
| stddev | List<number> | リスト内のすべての数値の標準偏差を取得します。 | number |
| flatten | List<List<?>> | ネストされたリストのコレクションを、すべての要素を含む単一のリストに変換します。 | List<?> |
| set_intersection | List<List<string>> | コレクションのすべての文字列リストで見つかった文字列のリストを取得します。 | List<string> |

| 集約 | 入力 | 意味 | 出力 |
| --- | --- | --- | --- |
| All above | List<List<?>> | ネストされたリストに対するすべてのオペレーションをサブリスト別に適用し、結果のリストを生成します。 | List<?> |

次の表は、集約関数を使用する有効なプロパティ式を示しています。

| 式 | 意味 | 結果のタイプ |
| --- | --- | --- |
| flatten(teams[*].players.attributes[skill]) | マッチングに属するすべてのプレイヤーの「skill」属性 (グループ分けしない) | List<number> |
| avg(teams[red].players.attributes[skill]) | Red チームに属するすべてのプレイヤーの平均スキル | number |
| avg(teams[*].players.attributes[skill] | マッチングの各チームの平均スキル | List<number> |
| avg(flatten(teams[*].players.attributes[skill])) | マッチングに属するすべてのプレイヤーの平均スキルレベル。この式では、プレイヤースキルのフラット化されたリストを取得し、スキルを平均化します。 | number |
| count(teams[red].players) | Red チームのプレイヤーの数 | number |

| 式 | 意味 | 結果のタイプ |
| --- | --- | --- |
| count (teams[*].players) | マッチングのチーム別のプレイヤー数 | List<number> |
| max(avg(teams[*].players.attributes[skill])) | マッチングの最高のチームスキルレベル | number |

# FlexMatch マッチメーキングイベント

Amazon GameLift FlexMatch は、処理されるマッチメーキングチケットごとにイベントを発行します。「FlexMatch イベント通知をセットアップする」の説明に従って、これらのイベントを Amazon SNS トピックに発行できます。これらのイベントは、ほぼリアルタイムで、ベストエフォートベースで Amazon CloudWatch Events にも放出されます。

このトピックでは、FlexMatch イベントの構造について説明し、各イベントタイプの例を提供します。マッチメーキングチケットのステータスの追加情報については、[Amazon GameLift API Reference] ( Amazon GameLift API リファレンス ) の[MatchmakingTicket](MatchmakingTicket)を参照してください。

## MatchmakingSearching

チケットはマッチメーキングに入力済みです。これには、新しいリクエストと、失敗したマッチング案の一部であるリクエストが含まれます。

[Resource: ](リソース:) ConfigurationArn

[Detail: ](詳細:) type、tickets、estimatedWaitMillis、gameSessionInfo

### 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

マッチング候補が作成済みです。これは、承諾が必要かどうかに関係なく、すべての新しい潜在的なマッチングに対して発行されます。

[Resource:](リソース:) ConfigurationArn

[Detail:](詳細:)
type、tickets、acceptanceTimeout、acceptanceRequired、ruleEvaluationMetrics、gameSessionInfo、matc

### 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

プレイヤーがマッチング候補を承諾済みです。このイベントは、マッチングに含まれる各プレイヤーの現在の承諾ステータスを示します。データが欠落している場合、そのプレイヤーに対しては AcceptMatch が呼び出されていないことを示します。

[Resource:](リソース:)ConfigurationArn

[Detail:](詳細:) type、tickets、matchId、gameSessionInfo

### 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

プレイヤーの承諾/却下または承諾のタイムアウトにより、マッチングの承諾プロセスが完了したことを示します。

[Resource:](リソース:)ConfigurationArn

[Detail:](詳細:) type、tickets、acceptance、matchId、gameSessionInfo

### 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

マッチメーキングが正常に完了し、ゲームセッションが作成済みです。

[Resource:](リソース:) ConfigurationArn

[Detail:](詳細:) type、tickets、matchId、gameSessionInfo

### 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

タイムアウトによってマッチメーキングチケットが失敗しました。

[Resource:](リソース:) ConfigurationArn

[Detail:](詳細:) type、tickets、ruleEvaluationMetrics、message、matchId、gameSessionInfo

### 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

マッチメーキングチケットがキャンセル済みです。

[Resource:](リソース:) ConfigurationArn

[Detail:](詳細:) type、tickets、ruleEvaluationMetrics、message、matchId、gameSessionInfo

## 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

マッチメーキングチケットでエラーが発生しました。ゲームセッションキューにアクセスできないか、内部エラーが発生した可能性があります。

[Resource:](リソース:) ConfigurationArn

[Detail:](詳細:) type、tickets、ruleEvaluationMetrics、message、matchId、gameSessionInfo

### 例

```
{
  "version": "0",
  "id": "025b55a4-41ac-4cf4-89d1-f2b3c6fd8f9d",
  "detail-type": "GameLift Matchmaking Event",
  "source": "aws.gamelift",
  "account": "123456789012",
  "time": "2017-08-16T18:41:09.970Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:gamelift:us-west-2:123456789012:matchmakingconfiguration/
SampleConfiguration"
```

```
  ],
  "detail": {
    "tickets": [
      {
        "ticketId": "ticket-1",
        "startTime": "2017-08-16T18:41:02.631Z",
        "players": [
          {
            "playerId": "player-1",
            "team": "red"
          }
        ]
      }
    ],
    "customEventData": "foo",
    "type": "MatchmakingFailed",
    "reason": "UNEXPECTED_ERROR",
    "message": "An unexpected error was encountered during match placing.",
    "gameSessionInfo": {
      "players": [
        {
          "playerId": "player-1",
          "team": "red"
        }
      ]
    },
    "matchId": "3ea83c13-218b-43a3-936e-135cc570cba7"
  }
}
```

# FlexMatch でのセキュリティ

AWS では、クラウドセキュリティを最優先事項としています。AWS のユーザーは、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを利用できます。

セキュリティは、AWS とユーザーの間の責任共有です。FlexMatch を使用する際に責任共有モデルを適用する方法の詳細については、Amazon GameLift のセキュリティ を参照してください。。

# Amazon GameLift FlexMatch リリースノートと SDK バージョン

Amazon GameLift リリースノートには、新しい FlexMatch に関する機能、更新、サービスに関する修正の詳細が記載されています。このページには Amazon GameLift SDK のバージョン履歴も含まれています。

# Amazon GameLift デベロッパーリソース

すべての Amazon GameLift ドキュメントとデベロッパーリソースを表示するには、Amazon GameLift ドキュメントのホームページを参照してください。

# AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。