アマゾン GameLift

FlexMatch 開発者ガイド

Version

アマゾン GameLift: FlexMatch 開発者ガイド

Table of Contents

アマゾンとは何ですか GameLiftFlexMatch?

GameLiftFlexMatchAmazon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イ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なものかを定義し、マッチ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するカスタムルールセットを構築できます。マッチングアルゴリズムの微調整など、マッチメイキングプロセスの重要な側面をゲームに合わせてカスタマイズすることもできます。

FlexMatchAmazon GameLift のゲームホスティングソリューション (リアルタイムサーバーを含む) としても、スタンドアロンのマッチメイキングサービスとしても利用できます。FlexMatchpeer-to-peerアーキテクチャを使用するゲームのスタンドアロン機能として実装することも、オンプレミスまたは他のクラウドコンピューティングソリューション (Amazon GameLift FleetIQ を含む) でゲームサーバーをホストすることもできます。このガイドでは、これらのシナリオのマッチメイ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ゲームの要件に応じてマッチメイキングの優先順位を柔軟に設定できます. 例えば、次の操作を実行できます。

- 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適したマッチを素早く見つけたり、プレイヤーにベストマッチを見つけるまでもう少し待たせたりして、プレイヤーにとって最適なプレイ体験を提供しましょう。
- 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。スキルや経験値など、すべてのプレイヤーが同じような特徴を持つマッチを作りましょう。あるいは、個々の選手の特性がより多様であっても、各チームの様々な特徴が類似する形で試合を形成します。
- 試合におけるプレイヤーのレイテンシー要素を優先順位付けします。マッチに参加しているすべてのプレイヤーにレイテンシーにハードリミットを設定するか、マッチに参加している全員が同じようなレイテンシーを経験するようにするか、あるいはその両方を行います。

FlexMatch作業を開始する準備はできましたか？

step-by-stepゲームを起動して実行するための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。

- FlexMatchAmazon GameLift ホスティングとの統合 (p. 8)
- GameLiftFlexMatchスタンドアロンマッチメイキング用の Amazon 統合 (p. 7)

FlexMatch主な機能

以下の機能は、スタンドアロンサービスとして使用する場合でも、Amazon FlexMatch GameLift ゲームホスティングとして使用する場合でも、FlexMatchすべてのシナリオで使用できます。

- [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イ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、FlexMatchマッチの新しいゲームセッションを開始する前に、カスタム承認ワークフローを開始し、プレイヤーの反応を報告してください。すべてのプレイヤーがマッチを受け入れたわけではない場合、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イ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必要に応じて他のプレイヤーを探してマッチを埋めるのに使えます。
- [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。ルール拡張により、最初のマッチルールを緩和する場所とタイミングを決めることができるため、プレイヤーはより早くプレイアブルゲームに参加できます。
- [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。新規プレイヤーをリクエストするタイミングや方法をカスタマイズしたり、同じカスタムマッチルールを使って他のプレイヤーを探したりできます。

FlexMatchGameLiftアマゾンホスティングで

Amazon FlexMatch でホストされているゲームにはGameLift、以下の追加機能があります。これらは、Amazon GameLift を使用してカスタムゲームサーバーをホストする場合、またはリアルタイムサーバーを使用する場合に使用できます。Amazon GameLift FleetIQ を使用して Amazon Elastic Compute Cloud (Amazon EC2) リソースでホストされているゲームは、FlexMatchスタンドアロン機能として実装する必要があります。

- [Game session placement] (ゲームセッションの配置) マッチが成功すると、FlexMatch Amazonに新しいゲームセッションプレースメントが自動的にリクエストされますGameLift。プレイヤーIDやチーム割り当てなど、マッチメイキング中に生成されたデータがゲームサーバーに提供され、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次に、ゲームセッション接続情報を戻して、ゲームクライアントがゲームに参加できるようにします。マッチ中のプレイヤーのレイテンシーを最小限に抑えるため、Amazon でのゲームセッションプレースメントでは、GameLift地域別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 (提供されている場合)。
- [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、プレーヤースロットが埋まっていない状態で新しいゲームセッションが開始されると、FlexMatch自動的にマッチバックフィルリクエストが送信されます。マッチメイキングシステムは、最小限のプレイヤー数でゲームセッションの配置プロセスを開始し、残りのスロットをすぐに埋めます。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自動バックフィルで入れ替えることはできません。

アマゾンの価格 GameLift FlexMatch

Amazon では、インスタンスは使用期間ごとに、GameLift帯域幅については転送されたデータ量に応じて課金します。Amazon GameLift のサーバーでゲームをホストする場合、FlexMatch使用量は Amazon GameLift の料金に含まれます。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 の料金と価格の詳細な一覧についてはGameLift、「Amazon GameLift 料金表」を参照してください。

Amazon でのゲームやマッチメイキングのホスティングにかかる費用の計算についてはGameLift、「Amazon GameLift 料金の見積もりの作成」を参照してください。使用方法が説明されています。AWS Pricing Calculator

GameLiftFlexMatchアマゾンの仕組み

このトピックでは、FlexMatchシステムのコアコンポーネントやそれらの相互作用など、Amazon GameLift FlexMatch サービスの概要を説明します。

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。FlexMatchGameLiftリアルタイムサーバーを含むAmazonサーバーでホストされているゲームは、GameLift統合されたAmazon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開始します。Amazon GameLift FleetIQ FlexMatch などのスタンドアロンサービスとして使用するゲームは、既存のホスティングシステムと連携してホスティングリソースを割り当て、試合のゲームセッションを開始する必要があります。

FlexMatchゲームの設定に関する詳細なガイダンスについては、を参照してくださいFlexMatch の開始方法 (p. 7)。

マッチメーキングコンポーネント

FlexMatchマッチメイ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.

GameLiftアマゾンのコンポーネント

これらは、FlexMatchサービスがゲームのマッチメイキングを行う方法を制御する Amazon GameLift リソースです。これらは、コンソールや AWS CLI などの Amazon GameLift ツールを使用して作成および管理するか、Amazon AWS SDK を使用してプログラム的に管理します。GameLift

- FlexMatchマッチメイキング設定 (マッチメーカーとも呼ばれます) — マッチメーカーは、ゲームのマッチメイ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。ゲームがマッチメイキングリクエストをに送信するとFlexMatch、どのマッチメーカーを使用するかが決まります。
- FlexMatchマッチメイキングルールセット — ルールセットには、マッチする可能性のあるプレイヤーを評価し、承認または拒否するために必要な情報がすべて含まれています。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のみ) — ゲームセッションキューは、利用可能なホスティングリソースを検索し、マッチの新しいゲームセッションを開始します。FlexMatchキューの設定によって、Amazon GameLift が利用可能なホスティングリソースを探す場所と、マッチングに最適なホストを選択する方法が決まります。

カスタムコンポーネント

以下のコンポーネントには、FlexMatch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システム全体に必要な機能が含まれています。

- [Player interface for matchmaking] (マッチメイ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イ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、マッチメイ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で行う必要があります。
- クライアントマッチメイキングサービス — このサービスは、プレイヤーインターフェースからのプレイヤー参加リクエストを処理し、FlexMatchマッチメイキングリクエストを生成してサービスに送信しま

す。処理中のリクエストについては、マッチメイキングイベントをモニタリングし、マッチメイ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は、AWS SDK と Amazon GameLift API FlexMatch を使用してサービスと通信します。

- 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探し、マッチの新しいゲームセッションを開始します。FlexMatchコンポーネントは、マッチメイ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マッチメイキングプロセス

このトピックでは、基本的なマッチメイキングのシナリオと、FlexMatchさまざまなゲームコンポーネントと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、FlexMatchクライアントのマッチメイキングサービスはマッチメイキングリクエストをに送信します. リクエストは、FlexMatchリクエストを処理する際に使用するマッチメーカーを特定します。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イキングリクエストを行うことができます。

マッチメイキング プールにリクエストを追加

FlexMatchマッチメーキングリクエストを受け取ると、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イキングサービスには、チケットステータスの変更など、マッチメイ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以下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) のマッチメイキングでは、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イキングイベントが生成されます。

- FlexMatchスタンドアロンサービスとして:マッチメイキングイベントが成功すると、ゲームにマッチ結果が届きます。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。

- FlexMatchAmazon GameLift ホスティングソリューションの場合:マッチ結果は自動的に Amazon GameLift キューに渡され、ゲームセッションが配置されます。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イキング結果データを使用できます。

- FlexMatchスタンドアロンサービスとして:カスタムマッチプレースメントサービスは、成功したマッチメイキングイベントのマッチ結果データを取得し、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て、マッチに利用可能なホスティングリソースを探します。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。
- FlexMatchAmazon GameLift ホスティングソリューションの場合: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探します。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実行されると、Amazon GameLift サービスはゲームサーバーに新しいゲームセッションを開始するように促し、マッチメイキングの結果やその他のオプションのゲームデータを渡します。

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。

- FlexMatchスタンドアロンサービスとして: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提供します。
- FlexMatchAmazon GameLift ホスティングソリューションの場合:ゲームセッションのプレースメントが成功すると、マッチしたすべてのチケットをゲームセッション接続情報とプレイヤーセッション ID FlexMatch で更新します。

FlexMatch の設定

GameLiftFlexMatchAWSAmazonはサービスであり、AWSこのサービスを利用するには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 FlexMatch GameLift ソリューションと併用している場合は、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のセットアップ
- Amazon 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設定する GameLift

Amazon のアカウントを設定するには GameLift

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 FlexMatch を呼び出してリソースにアクセスするすべてのユーザーに、明示的なアクセス権を与える必要があります。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 の場合はGameLift、カスタムアクセス権限ポリシーを作成し、それを各ユーザーまたはユーザーグループにアタッチ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に対する管理者権限を含むインラインアクセス権限ポリシーを示しています。FlexMatch特定の項目のみを指定してアクセスを制限することができます。

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

FlexMatch の開始方法

このセクションのリソースを参考にして、マッチメイキングシステムの構築を始めるのに役立ててください. FlexMatch

トピック

- GameLiftFlexMatchスタンドアロンマッチメイキング用の Amazon 統合 (p. 7)
- FlexMatchAmazon GameLift ホスティングとの統合 (p. 8)

GameLiftFlexMatchスタンドアロンマッチメイキング用の Amazon 統合

このトピックでは、FlexMatchスタンドアロンのマッチメイキングサービスとして実装するための完全な統合プロセスの概要を説明します. マルチプレイヤーゲームが、peer-to-peerカスタム構成の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は、このプロセスを使用してください。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している場合は、を参照してくださいFlexMatchAmazon GameLift ホスティングとの統合 (p. 8)。

統合を開始する前に、Amazon AWS GameLift サービスのアカウントを取得し、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 (p. 6)」を参照してください。Amazon GameLift FlexMatch マッチメーカーとルールセットの作成と管理に関連するすべての重要なタスクは、Amazon GameLift コンソールを使用して実行できます。

1. FlexMatchマッチメイキングルールセットを作成する. 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、マッチが有効であるために満たす必要のある一連の要件も指定できます。これによって、FlexMatch マッチにプレーヤーを含めるか除外するかが決まります。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。また、FlexMatchルールセットのアルゴリズムをカスタマイズして、最大200人のプレイヤーで大規模なマッチを組むこともできます。以下のトピックを参照してください。

   - FlexMatchルールセットを作成 (p. 12)
   - FlexMatchルールセットの例 (p. 22)
2. イベントの通知を設定します。通知を使用して、保留中のマッチリクエストのステータスなど、FlexMatchマッチメイキングのアクティビティを追跡できます。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。マッチメイ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イベント通知の設定 (p. 41)
   - FlexMatchマッチメイキングイベント (p. 74)
3. FlexMatchマッチメイキングの設定を行います。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マッチメーカーをデザインしよう (p. 10)
   - マッチメーキング設定の作成 (p. 38)

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。マッチメイキングリクエストを作成および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するFlexMatch. マッチメイ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、マッチメイ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。

   - FlexMatchゲームクライアントに追加 (p. 45)

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、マッチメイキングを使用して、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。

FlexMatchAmazon GameLift ホスティングとの統合

FlexMatchGameLiftカスタムゲームサーバーとリアルタイムサーバー用のマネージドAmazonホスティングで利用できます。FlexMatch マッチメーキングをゲームに追加するには、以下のタスクを行います。

- 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ルールセットを作成 (p. 12)
    - FlexMatchルールセットの例 (p. 22)
  - 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、マッチメイ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。
    - キューの作成
  - 通知を設定します (オプション)。マッチメイ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イベント通知の設定 (p. 41)
  - 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マッチメーカーをデザインしよう (p. 10)
    - マッチメーキング設定の作成 (p. 38)
- FlexMatch をゲームクライアントサービスに統合します。マッチメイ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。マッチメイ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。
  - FlexMatchゲームクライアントに追加 (p. 45)
- FlexMatch をゲームサーバーに統合します。マッチメイ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 FlexMatch GameLift がホストするゲームサーバーに追加する (p. 50)

- 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オンにすると、Amazon GameLift にバックフィルリクエストを管理させることができます。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。
  - 既存のゲームをバックフィルして FlexMatch (p. 52)

    Note

    FlexMatchバックフィルは現在、リアルタイムサーバーを使用するゲームでは使用できません。

アマゾンの仲人の構築 GameLift FlexMatch

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 マッチメーカーでマッチメーキングリクエストを受け取って処理する方法の詳細については、「FlexMatchマッチメイキングプロセス (p. 4)」を参照してください。

トピック

FlexMatchマッチメーカーをデザインしよう

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ルールセットを作成 (p. 12)」および「FlexMatchルールセットの例 (p. 22)」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イ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イベント通知の設定 (p. 41)」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 FlexMatch GameLift マネージドホスティングで使用する場合、ゲームセッションキューはマッチ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て、新しいゲームセッションを開始します。各キューは、ゲームセッションを配置できる場所を決定する場所とリソースタイプ (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) のリストで構成されます。キューの詳細については、「マルチロケーションキューの使用」を参照してください。

マッチメーカーの場所を選択してください

マッチメイキングを行う場所を決めて、その場所にマッチメーカー (マッチメイキング設定とルールセット) を作成します。マッチメーカーに対するすべてのリクエストは、チケット プールに送信され、そこでソートされ、実行可能な試合が評価されます。マッチが行われた後、プレイヤー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のゲームセッションに誘導されます。

マッチメーカーの口ケーションを選ぶ際には、口ケーションがプレイヤーのパフォーマンスにどのような影響を与えるか、また対象となるプレイヤーのマッチ体験を最適化する方法を検討してください。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、FlexMatchプレイヤーとマッチメーキングリクエストを送信するクライアントサービスに近い場所に配置します。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。
- ゲームが世界中の視聴者に届く場合は、複数の場所にマッチメーカーを作成し、マッチリクエストをプレイヤーに最も近いマッチメーカーに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 FlexMatch GameLift マネージドホスティングで使用する場合は、マッチメーカーとマッチメーカーが使用するゲームセッションキューを同じ場所に配置してください。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 FlexMatch GameLift ホスティングソリューション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。FlexMatchスタンドアロンのマッチメイキングサービス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をお勧めします。

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、マッチメイキング 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すると、ゲームセッションの全期間にわたって、マッチした新規プレイヤーでゲームセッションを埋め尽くすことができます。バックフィルリクエストを処理する際には、FlexMatch元のプレイヤーとのマッチングに使用されたものと同じマッチメーカーを使用します。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 FlexMatch GameLift ホスティングを使用するか、他のホスティングソリューションを使用するかにかかわらず利用できます。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利用できます。この機能を有効にすると、空いているプレイヤースロットでゲームセッションが開始されると、Amazon GameLift はそのスロットに対するバックフィルリクエストを自動的に生成し始めます。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イキング プールに入力するとすぐに補充されるようにマッチメイ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 FlexMatch GameLift マネージドホスティングを使用するゲームでは、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指定できます。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。異なるマッチメイ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ルールセットを作成

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 のいずれかを使用できます。

トピック

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人のプレイヤーが参加する小規模マッチ用のルールセットを作成する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には 2 つの役割があります。1 つはマッチのチーム構成と規模を定めること、もう 1 つはマッチメーカーにプレイヤーの選び方を伝え、可能な限りベストなマッチを組むことです。

しかし、マッチメイキングのルールセットにはさらに多くの機能があります. 例えば、以下のことが可能です。

- ゲームに合わせてマッチメイキングアルゴリズムを最適化.
- ゲームプレイの質を守るために、プレイヤーのレイテンシーの最小要件を設定します。
- 時間をかけてチーム要件やマッチルールを徐々に緩和し、すべてのアクティブプレイヤーが希望するマッチを見つけられるようにします。

- パーティーアグリゲーションを使用してグループのマッチメイ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の大規模マッチを処理します。大規模マッチの作成の詳細については、を参照してくださいFlexMatchマッチ率の高いルールセットのデザイン (p. 18)。

マッチメイキングルールセットを作成する際には、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。

- ルールセットの説明 (必須) (p. 13)
- マッチアルゴリズムのカスタマイズ (p. 13)
- プレイヤー属性の宣言 (p. 16)
- 対戦チームの定義 (p. 16)
- プレイヤーマッチングのルール設定 (p. 17)
- 時間の経過による要件の許可 (p. 17)

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

ルールセットの説明 (必須)

ルールセットの詳細を指定します。

- 名前 (オプション) — 自分用に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したルールセット名とは関係ありません。
- ruleLanguageVersion— FlexMatch ルールの作成に使用されるプロパティ表現言語のバージョン。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ほとんどのゲームのデフォルトアルゴリズムを最適化し、プレイヤーを最小限の待ち時間で受け入れられるようにしています。アルゴリズムをカスタマイズして、ゲームに合わせてマッチメイキングを調整できます.

FlexMatch以下はデフォルトのマッチメイキングアルゴリズムです:

1. FlexMatch開いているすべてのマッチメイキングチケットとバックフィルチケットをチケットプールに入れます.
2. FlexMatch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につれて、FlexMatch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各バッチ内のチケットを年齢別にソートします。
4. FlexMatch各バッチの最も古いチケットに基づいてマッチを構築します。

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 (p. 60) コマンドのリファレンスを参照してください。

以下のオプションカスタマイズを使用して、マッチメイキングプロセスのさまざまな段階に影響を与えましょう。

- 事前バッチソートの追加 (p. 14)
- BatchDistance 属性に基づくフォームバッチ
- バックフィルチケットの優先順位付け (p. 15)
- 拡張時に古いチケットを優先 (p. 15)

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できます。このタイプのカスタマイズは、チケットプールの多いゲームで最も効果的です。事前にバッチをソートしておくと、マッチメイキングのプロセスをスピードアップし、定義された特性におけるプレイヤーの統一性を高めることができます。

アルゴリズムプロパティを使用してバッチ前のソート方法を定義します。batchingPreferenceデフォルトの設定は、random です。

バッチ前ソートのカスタマイズオプションには次のものがあります。

- [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に分類するために、プレイヤー属性のリストを提供してください。

  プレーヤー属性別にソートするには、batchingPreferenceに設定しsorted、でプレーヤー属性のリストを定義しますsortByAttributes。属性を使用するには、playerAttributesまずルールセットのコンポーネントで属性を宣言します。

  次の例では、FlexMatchプレイヤーの好みのゲームマップに基づいてチケットプールをソートし、次にプレイヤースキルでソートします。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) 可能な限り低いレイテンシーでマッチを作成することも、許容できるレイテンシーでマッチをすばやく作成することもできます。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。

  strategyアルゴリズムのプロパティをに設定しますbalanced。バランスの取れた戦略では、使用可能な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、「FlexMatchマッチ率の高いルールセットのデザイン (p. 18)」を参照してください。

  FlexMatch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、次のいずれかの方法で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、プレイヤーが最も低いレイテンシー値を報告した場所別に事前にソートされています。FlexMatch次に、同じ場所で低レイテンシーでチケットをバッチ処理し、より良いゲームプレイ体験を実現します。また、各バッチのチケット数が減るため、マッチメイキングに時間がかかる場合があります。batchingPreferenceこのカスタマイズを使用するにはfastestRegion、次の例のようにに設定します。

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

  - 許容範囲のレイテンシーはすばやく一致します。チケットプールは、プレイヤーが許容可能なレイテンシー値を報告した場所別に事前に分類されています。これにより、より多くのチケットを含むバッチの数が少なくなります。各バッチのチケット数が多いほど、一致するチケットをより早く見つけることができます。このカスタマイズを使用するには、batchingPreference largestPopulation次の例に示すようにプロパティをに設定します。

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

Note

バランス戦略のデフォルト値はです`largestPopulation`。

バックフィルチケットの優先順位付け

ゲームに自動バックフィルまたは手動バックフィルが実装されている場合は、FlexMatchリクエストタイプに基づいてマッチメイキングチケットの処理方法をカスタマイズできます。リクエストタイプは、新規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両方のタイプのリクエストを同じように扱います。

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

FlexMatch複数のバックフィルチケットを一緒にマッチさせることはできません。

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

- 最初にバックフィルチケットをマッチさせてください。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、新規プレイヤーは既存のゲームに参加する可能性が高いということです。

  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、ゲームセッションが短く、プレイ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、これらのゲームが最低限のマッチを組み、FlexMatch空いているスロットを埋めるためにさらに多くのプレイヤーを探しながらゲームを始めるのに役立ちます。

  `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させてください。このオプションは、他のすべてのチケットを評価するまでバックフィルチケットを無視します。つまり、入ってく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、FlexMatch既存のゲームにバックフィルします。

  このオプションは、新しいマッチを組むのに十分な数のプレーヤーがいない場合など、プレーヤーをゲームに参加させる最後のチャンスとしてバックフィルを使用したい場合に便利です。

  `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、マッチを完了するのが難しい場合のマッチ条件を緩和します。Amazonでは、部分的に完了したマッチのチケットが特定の年齢に達すると、GameLift拡張ルールが適用されます。チケットの作成タイムスタンプによって、GameLift Amazon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最後に一致したチケットのタイムスタンプを追跡します。

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

- 最新のチケットに基づいて拡張できます。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新しいチケットと一致するたびに、タイムクロックはリセットされます。このオプションを使用すると、結果のマッチの質は高くなる傾向がありますが、マッチには時間がかかります。マッチに時間がかかりすぎると、マッチリクエストが完了する前にタイムアウトする可能性があります。expansionAgeSelectionに設定しますnewest。 newestがデフォルトです。
- 最も古いチケットに基づいて拡張できます。このオプションは、一致する可能性のあるチケットの中で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拡張の適用が速くなり、最も早くマッチしたプレイヤーの待ち時間が改善されますが、すべてのプレイヤーのマッチの質は低下します。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、マッチメイキングリクエストに含める個々のプレイヤー属性を一覧表示します。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の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とき。たとえば、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。有効なデータ型は、数値、文字列、または文字列マップです。
- デフォルト (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に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、小規模なマッチング、または大規模なマッチングとしてマッチングリクエストを処理します。最大40人のプレイヤーがマッチする可能性のあるマッチはスモールマッチで、40人以上のプレイヤーが参加するマッチは大規模マッチです。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、ルールや拡張パック、FlexMatch ゲームセッションでのマッチメイキングデータの参照に使用します。
- MaxPlayers (必須) — チームに割り当てるプレー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ーヤーの最小数を指定します。
- 数量 (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と番号が付加されます。たとえばRed-Team1、Red-Team2、およびRed-Team3。

FlexMatchチームを最大人数のプレーヤーに詰め込もうとしますが、プレーヤーの数が少ない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ルールの例については、FlexMatchルールセットの例 (p. 22)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

ルールステートメントを作成して、プレーヤーがマッチに参加できるかどうかを評価します。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。GameLiftAmazonがマッチリクエストを処理する際、利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作成します。FlexMatchルール作成の詳細なヘルプについては、を参照してくださいFlexMatchルールタイプ (p. 67)。

- 名前 (必須) — ルールセット内のルールを一意に識別するわかりやすい名前。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 (p. 59)」を参照してください。
- ルールタイププロパティ (必須の場合もあります) — 定義されているルールのタイプによっては、特定のルールプロパティを設定する必要がある場合があります。プロパティと FlexMatch プロパティ式言語の使用方法については、「FlexMatchルール言語 (p. 59)」を参照してください。

時間の経過による要件の許可

拡張機能を使用すると、一致するものが見つからなかった場合に、FlexMatch徐々にルール条件を緩和できます。FlexMatchこの機能により、完全に一致しない場合でもベストアベイラビリティが確保されます。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年齢が拡張待ち時間と一致したときに開始されます。FlexMatchマッチに新しいチケットを追加すると、拡張待ち時間クロックがリセットされることがあります。algorithmルールセットのセクションで拡張の開始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。ルールセットにはディスタンスルールステートメントが使用されています。これは、SkillDeltaマッチに参加するすべてのプレーヤーが互いのスキルレベルが5レベル以内であることを要求するように命名されたものです。15秒間新たなマッチが行われなかった場合、この拡張パックではスキルレベルの差が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。より良いアプローチは、ゲームの自動バックフィルが開始される傾向がある後に拡張を開始することです。拡張のタイミングはチーム構成によって異なるため、テストを行ってゲームに最適な拡張戦略を見つけてください。

FlexMatchマッチ率の高いルールセットのデザイン

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、一般的なマッチメイ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 (p. 61)」およびルールセットの例「例 7: ラージ試合を作成する (p. 32)」 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。この設定によりFlexMatch、プロパティで定義されている特定のプレーヤー属性に基づいて最適なチームバランスを見つけるために、試合後に追加のチェックが行われます。`balancedAttribute`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取る場合は、FlexMatchすべてのチームのスキルレベルができるだけ均等になるようにプレーヤーを割り当てるようにします。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プール内のチケットのうち、少なくとも1つの場所で許容可能なレイテンシー値を持つ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、レイテンシーが許容範囲内であっても、常に最適であるとは限らないゲームに配置されることがあります。`batchingPreference`プロパティが設定されていない場合、が「バランス」`strategy` に設定されているときのデフォルトの動作です。
  - 最速の口ケーション。FlexMatchプール内のすべてのチケットを、最も低いレイテンシー値が報告された場所に基づいて事前にソートします。そのため、同じ口ケーションでレイテンシーが低いと回答したプレイヤーと試合が組まれる傾向にあります。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、レイテンシーの優先度が高いため、試合に参加するプレイヤーのバランシング属性は大きく異なる可能性があります。

次の例では、マッチアルゴリズムが次のように動作するように構成しています。 (1) チケットプールを事前にソートして、許容できる待ち時間値のある場所でチケットをグループ化します。 (2) ソートされたチケットをバッチ作成してマッチングし、 (3) チケットを含むマッチをバッチで作成し、チームのバランスを取り、平均的なプレイヤースキルを均等にします。

```
"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、マッチメイ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マッチングと同様ですが、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。それに応じて、チームの最小人数を調整したい場合があります。

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. すべてのチームが揃った状態でminPlayers、空いているスロットの数に基づいて追加のプレーヤーが割り当てられ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。

大規模マッチのルールを設定

ラージな対戦のマッチメイキングは、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、次の種類のルールを組み込むことができます。

- プレイヤーのレイテンシーにハードリミットを設定するルール。latencyプロパティにはルールタイプを使用してくださいmaxLatency。レイテンシールール (p. 70)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
    }],
```

- 指定され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いた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イキングチケットをバッチ処理します。そのため、指定された属性で一致するプレイヤー同士のマッチングにつながる傾向があります。batchDistanceルールタイプを使用し、数値ベースの属性を識別し、許容範囲を最も広く指定してください。バッチ距離ルール (p. 67)リファレンスを参照してください。マッチに参加するプレイヤー同士のスキルレベルを一定に保つ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ゲームセッションの開始後にのみバックフィルリクエストの生成を開始しますが、マッチが作成されてから数秒間は発生しない場合があります。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 GameLift FlexMatch マッチメーカーのマッチメイキングルールセットを作成する前に、ルールセットの構文を確認することをお勧めします (p. 59)。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リージョンに含めることが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 (p. 12)
- FlexMatchルールセットの例 (p. 22)
- FlexMatchルール言語 (p. 59)

Console

ルールセットの作成

1. https://console.aws.amazon.com/gamelift/ で Amazon GameLift コンソールを開きます。
2. AWSルールセットを作成するリージョンに切り替えます。ルールセットを使用するマッチメイキング設定と同じリージョンでルールセットを定義します。
3. ナビゲーションペインでFlexMatch、「マッチメイキングルールセット」を選択します。
4. 「マッチメイキングのルールセット」ページで、「ルールセットの作成」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、メトリック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、ルールセットを JSON で入力します。ルールセットの設計については、を参照してくださいFlexMatchルールセットの設計 (p. 12)。のサンプルルールセットのいずれかを使用することもできますFlexMatchルールセットの例 (p. 22)。
   c. [検証] を選択して、ルールセットの構文が正しいことを確認します。ルールセットは作成後に編集できないため、最初にルールセットを検証することをお勧めします。
   d. (オプション) [タグ] に、AWSリソースの管理と追跡に役立つタグを追加します。
6. [作成] を選択します。作成が成功すると、マッチメーカーでルールセットを使用できます。

AWS CLI

ルールセットの作成

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。

このコマンド例は、1 つのチームを設定するシンプルなマッチメイキングルールセットを作成します。ルールセットは、AWS必ずそれを使用するマッチメイキング設定と同じリージョンに作成してください。

```
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 GameLift MatchmakingRuleSetは指定した設定を含むオブジェクトを返します。マッチメーカーは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、「マッチメイキングルールセット」を選択します。
4. 「マッチメイ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] ダイアログボックスで、[削除] を選択して削除を確定します。

Note

マッチメイキング設定がルールセットを使用している場合、Amazonはエラーメッセージ (ルールセットを削除できません) GameLiftを表示します。このような場合は、別のルールセットを使用するようにマッチメイキング設定を変更してから、もう一度試してください。どのマッチメイキング設定がルールセットを使用しているかを確認するには、ルールセットの名前を選択して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、delete-matchmaking-rule-setコマンドを使用してマッチメイキングルールセットを削除します。

マッチメイキング設定がルールセットを使用している場合、GameLift Amazonはエラーメッセージを返します。このような場合は、別のルールセットを使用するようにマッチメイ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イキング設定のリストを取得するにはdescribe-matchmaking-configurations、コマンドを使用してルールセット名を指定します。

このコマンド例では、マッチメイキングルールセットの使用状況を確認してから、ルールセットを削除します。

```
aws gamelift describe-matchmaking-configurations \
    --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。以下の例は、FlexMatch 設定構造およびプロパティ式の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 ルールおよびルールセットの詳しい使用方法については、以下のトピックを参照してください。

- FlexMatchルールセットを作成 (p. 12)
- FlexMatchルールセットの設計 (p. 12)
- FlexMatchルールセットスキーマ (p. 60)
- FlexMatchルール言語 (p. 59)

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。このシナリオでは、以下の手順に従います。

- 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、FlexMatch が暫定的にプレイヤー候補をチームに追加し、チーム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。
  - マッチメイ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。このシナリオは、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、マッチに参加しているすべてのプレイヤーが対戦相手となる free-for-all「」スタイルです。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文字) が、基準値の少なくとも1つの文字列 (プレーヤーが好む対戦相手) GameLiftと交差することを確認します。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、「FlexMatchマッチ率の高いルールセットのデザイン (p. 18)」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 (p. 20)」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 FlexMatch がこのルールセットに対してマッチングを満たす方法は、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 (p. 20)」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、同じような属性または同じような選択をしている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このルールは、次の条件の少なくとも 1 つが当てはまる場合に一致することを保証します。
  - mapmodeマッチに参加しているプレイヤーが同じものをリクエストしています。
  - マッチに参加しているプレイヤーには、skillleagueそれに匹敵する能力があります。

```
{
    "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: プレイヤー回避リスト

このルール例では、プレイヤーはマッチさせたくない他のプレイヤーを避けることができます。各プレイヤーは、マッチメーカーがプレイヤー選択時に評価する回避リストを定義できます。プレイヤーが既存のプレイヤーの回避リストに載っている場合、そのプレイヤーはマッチからブロックされます。

```
{
```

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

マッチメーキング設定の作成

マッチメイキングリクエストを処理するように Amazon GameLift FlexMatch マッチメーカーを設定するには、マッチメイ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) のいずれかを使用してください。マッチメーカーの作成の詳細については、を参照してくださいFlexMatchマッチメーカーをデザインしよう (p. 10)。

Amazon GameLift ホスティング用のマッチメーカーを作成

マッチメイ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と Amazon GameLift ゲームセッションキューを作成します (p. 20)。

Console

1. Amazon GameLift コンソールのナビゲーションペインで、「マッチメイキング設定」を選択します。
2. AWSマッチメイカーを作成したいリージョンに切り替えます。
3. 「マッチメイキング設定」ページで、「マッチメイキング設定を作成」を選択します。
4. 「設定詳細の定義」ページの「マッチメイキング設定の詳細」で、次の操作を行います。

    a. [名前] には、リストや指標で識別しやすいマッチメーカー名を入力します。マッチメーカー名はリージョン内で一意でなければなりません。マッチメイキングリクエストでは、名前と地域によってどのマッチメーカーを使用するかが決まります.
    b. (オプション) [説明] に、マッチメーカーを識別しやすい説明を追加します。
    c. ルールセットでは、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。リストには、現在のリージョンで作成したすべてのルールセットが含まれます。
    d. FlexMatchモードについては、Amazon マネージドホスティングの「GameLiftマネージド」を選択します。このモードでは、FlexMatch成功したマッチを指定されたゲームセッションキューに渡すように求められます。

e. 「AWSリージョン」で、マッチメーカーで使用したい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、マッチメーカーで使用したいゲームセッションキューを選択します。

5. [Next] (次へ) を選択します。

6. 設定ページの「マッチメイキング設定」 で、次の操作を行います。

a. 「リクエストタイムアウト」には、マッチメーカーがリクエストごとにマッチを完了するまでの最大時間を秒単位で設定します。FlexMatchこの時間を超えるマッチメイ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オンにするには、「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か、バックフィル機能を使用しない場合は、「手動」を選択してください。

c. (オプション)「追加プレイヤー数」で、マッチで開いたままにするプレイヤースロットの数を設定します。FlexMatch将来、これらのスロットをプレイヤーでいっぱいにすることができます。

d. (オプション)「マッチ承認オプション」の「承認が必要」で、提案されたマッチに参加する各プレイヤーにマッチへの参加を積極的に受け付けるようにするには、「必須」を選択します。このオプションを選択した場合、「承認タイムアウト」で、マッチメーカーがマッチをキャンセルする前にプレイヤーの承認を待つ時間を秒単位で設定します。

7. (オプション) [イベント通知設定] で、次の操作を行います。

a. (オプション) SNS トピックでは、マッチメイキングイベントの通知を受信するための Amazon 簡易通知サービス (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、後でマッチメイキング設定を編集して選択できます。詳細については、「FlexMatchイベント通知の設定 (p. 41)」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には、イベントメッセージでこのマッチメーカーに関連付けるカスタムデータを入力します。FlexMatchこのデータをマッチメーカーに関連するすべてのイベントに含めます。

8. (オプション)「その他のゲームデータ」を展開してから、次の操作を行います。

a. (オプション) ゲームセッションデータには、FlexMatchこのマッチメイキング設定を使用して行われたマッチで開始された新しいゲームセッションに配信したい追加のゲーム関連情報を入力します。

b. (オプション) ゲームのプロパティには、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、AWSリソースの管理と追跡に役立つタグを追加します。

10. [Next] (次へ) を選択します。

11. [確認と作成] ページで、選択内容を確認し、[作成] を選択します。作成が完了すると、マッチメイカーはマッチメイキングリクエストを受け入れる準備が整います。

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。

このコマンド例は、プレイヤーの承認を必要とする新しいマッチメイキング設定を作成し、自動バックフィルを有効にします。また、FlexMatch後でプレーヤーを追加できるように 2 つのプレイヤースロットを確保し、一部の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"
```

マッチメイキング設定の作成リクエストが成功すると、Amazon GameLift MatchmakingConfigurationはあなたがマッチメーカーにリクエストした設定を含むオブジェクトを返します。新しいマッチメーカーは、マッチメイキングリクエストを受け入れる準備ができています.

スタンドアロン用のマッチメーカーを作成 FlexMatch

マッチメイ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ます (p. 20)。

Console

1. https://console.aws.amazon.com/gamelift/home で Amazon GameLift コンソールを開きます。
2. AWSマッチメイカーを作成したいリージョンに切り替えます。FlexMatchマッチメイキング設定をサポートするリージョンのリストについては、を参照してくださいマッチメーカーの場所を選択してください (p. 10)。
3. ナビゲーションペインでFlexMatch、「マッチメイキング設定」を選択します。
4. 「マッチメイキング設定」ページで、「マッチメイキング設定を作成」を選択します。
5. 「設定詳細の定義」ページの「マッチメイキング設定の詳細」で、次の操作を行います。

   a. [名前] には、リストや指標で識別しやすいマッチメーカー名を入力します。マッチメーカー名はリージョン内で一意でなければなりません。マッチメイキングリクエストでは、名前と地域によってどのマッチメーカーを使用するかが決まります.
   b. (オプション) [説明] に、マッチメーカーを識別しやすい説明を追加します。
   c. ルールセットでは、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。リストには、現在のリージョンで作成したすべてのルールセットが含まれます。
   d. FlexMatchモードについては、「スタンドアロン」を選択します。これは、Amazon 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GameLift。
6. [Next] (次へ) を選択します。
7. 設定ページの「マッチメイキング設定」 で、次の操作を行います。

   a. 「リクエストタイムアウト」には、マッチメーカーがリクエストごとにマッチを完了するまでの最大時間を秒単位で設定します。この時間を超えるマッチメイキングリクエストは拒否されます.
   b. (オプション)「マッチ承認オプション」の「承認が必要」で、提案されたマッチに参加する各プレイヤーにマッチへの参加を積極的に受け付けるようにするには、「必須」を選択します。このオプションを選択した場合、「承認タイムアウト」で、マッチメーカーがマッチをキャンセルする前にプレイヤーの承認を待つ時間を秒単位で設定します。
8. (オプション) [イベント通知設定] で、次の操作を行います。

   a. (オプション)「SNS トピック」で、マッチメイキングイベントの通知を受け取る 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、後でマッチメイキング設定を編集して選択できます。詳細については、「FlexMatchイベント通知の設定 (p. 41)」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には、イベントメッセージでこのマッチメーカーに関連付けるカスタムデータを入力します。FlexMatchこのデータをマッチメーカーに関連するすべてのイベントに含めます。

9. (オプション) [タグ] に、AWSリソースの管理と追跡に役立つタグを追加します。
10. [Next] (次へ) を選択します。
11. [確認と作成] ページで、選択内容を確認し、[作成] を選択します。作成が完了すると、マッチメイカーはマッチメイキングリクエストを受け入れる準備が整います。

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。

このコマンド例は、プレイヤーの承認を必要とするスタンドアロンのマッチメーカー用の新しいマッチメイ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"
```

マッチメイキング設定の作成リクエストが成功すると、Amazon GameLift MatchmakingConfigurationはあなたがマッチメーカーにリクエストした設定を含むオブジェクトを返します。新しいマッチメーカーは、マッチメイキングリクエストを受け入れる準備ができています.

マッチメイキングの設定を編集する

マッチメイ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イ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、以下の理由により既存の有効なマッチメーキングチケットがある場合、新しいルールセットに互換性がない可能性があります。

- 新規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用して新しいマッチメイキング設定を作成します。

FlexMatchイベント通知の設定

イベント通知を使用して、個々のマッチメイキングリクエストのステータスを追跡できます。プロダクションまたはマッチメイキングのアクティビティが多いプリプロダクションのゲームでは、すべてイベント通知を使用してください。

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 シンプル通知サービス (Amazon SNS) トピックにイベント通知を公開してもらいます。
- イベントを管理するには、EventBridge自動的に公開されるAmazonイベントとそのツール一式を使用してください。

Amazon FlexMatch GameLift が生成するイベントのリストについては、を参照してくださいFlexMatch マッチメイキングイベント (p. 74)。

EventBridgeイベントのセットアップ

GameLiftEventBridgeAmazonはすべてのマッチメイキングイベントをAmazonに自動的に投稿します. を使用するとEventBridge、マッチメーキングイベントをターゲットにルーティングして処理するようにルールを設定できます。たとえば、イベント "PotentialMatchCreated" AWS Lambda をプレイヤーの承認を処理する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 とはEventBridge?」を参照してください。

> Note
>
> マッチメーカーを設定するときは、通知対象フィールドを空白のままにするか、Amazon SNS EventBridge の両方を使用する場合は SNS トピックを参照してください。

Amazon SNS トピックをセットアップする

GameLiftFlexMatchマッチメーカーが生成するすべてのイベントを Amazon SNS トピックに公開するように設定できます。

Amazon GameLift イベント通知用の 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 AWS Management Console開発者ガイドの「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。 (わかりやすくするために、ポリシー全体をここに示しています。) 独自の SNS トピックと Amazon GameLift マッチメイキングの設定には、必ず Amazon リソースネーム (ARN) の詳細を使用してください。

```
{
  "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. [変更を保存] をクリックします。

サーバー側の暗号化による SNS トピックの設定

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存できます。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 Notification Service 開発者ガイドの「保存中の暗号化」を参照してください。

サーバー側の暗号化を使用する SNS トピックを設定するには、以下のトピックを確認してください。

- AWS Key Management Service開発者ガイドのキーの作成
- Amazon 簡易通知サービス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てください。

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

Lambda 関数を呼び出すためのトピックサブスクリプションの設定

Amazon SNS トピックに公開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。マッチメーカーを設定するときは、通知先を SNS トピックの ARN に設定してください。

AWS CloudFormation次のテンプレートは、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、MyFlexMatchEventTopicという名前の Lambda 関数を呼び出します。FlexMatchEventHandlerLambdaFunctionこのテンプレートは、Amazon が SNS GameLift トピックに書き込むことを許可する IAM アクセス権限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 関数を呼び出す権限を追加します。

```
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
```

ゲームの準備 FlexMatch

GameLiftFlexMatchAmazonを使用して、プレイヤーのマッチメイキング機能をゲームに追加してください. FlexMatchGameLiftカスタムゲームサーバーとリアルタイムサーバー用のマネージドAmazonソリューションで利用できます。

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、ゲームの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間のマッチング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イキングの詳細については、「FlexMatchルールセットの例 (p. 22)」を参照してください。

FlexMatch ではキュー機能を活用できます。マッチングを作成すると、マッチングの詳細が FlexMatch から指定の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ティングリソースを検索し、マッチ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イ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、「アマゾンの仲人の構築 GameLift FlexMatch (p. 10)」を参照してください。FlexMatch の仕組みの詳細については、「GameLiftFlexMatchアマゾンの仕組み (p. 3)」を参照してください。

FlexMatchゲームクライアントに追加

このトピックでは、FlexMatchクライアント側のゲームサービスにマッチメイキングサポートを追加する方法について説明します。GameLiftAmazonマネージドホスティングでも別のホスティングソリューションでも、プロセスは基本的に同じです。FlexMatchFlexMatch 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については、以下のトピックを参照してください。

- FlexMatchAmazon GameLift ホスティングとの統合 (p. 8)
- GameLiftFlexMatchアマゾンの仕組み (p. 3)
- アマゾンの仲人の構築 GameLift FlexMatch (p. 10)
- FlexMatchルールセットの例 (p. 22)

FlexMatchゲームでマッチメイキングを有効にするには、以下の機能を追加してください。

- 1人以上のプレイヤー（必要な）のマッチメイ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イ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、これはマッチメイ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。

- アマゾン GameLift API を追加します。クライアントサービスは、AWS SDK の一部である Amazon GameLift API の機能を使用します。GameLiftSDK の詳細と最新バージョンのダウンロードについては、「クライアントサービス用 Amazon AWS 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] (マッチメイキングチケットシステムのセットアップ) すべてのマッチメイ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イ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、「既存のゲームをバックフィルして FlexMatch (p. 52)」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イキングリクエストを管理します。FlexMatchマッチメイキングをリクエストするプロセスは、FlexMatch GameLiftmanaged FlexMatch Amazonホスティング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使用するゲームで同じです。

マッチメイキングリクエストを作成します。

- アマゾン GameLift API を呼び出しますStartMatchmaking。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指定されたマッチメーカーのプールに各リクエストを配置し、そのリクエストはマッチメーカーの設定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マッチメーカーをデザインしよう (p. 10)」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、マッチメイ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 つのマッチを作成し、すべてのプレイヤーを同じチーム (ランダムに選択) に割り当てようとします。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イ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イ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、マッチメイ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ルールセットを作成 (p. 12)」および「FlexMatchルールセットの例 (p. 22)」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。

- マッチリクエストが送信されたら、リクエストのチケット ID DescribeMatchmakingを使って呼び出すことでリクエストの詳細を確認できます。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。

- StopMatchmakingリクエストのチケットIDで電話することで、いつでもマッチメイキングリクエストをキャンセルできます.

マッチメイキングイベントの追跡

通知を設定して、GameLift Amazonがマッチメイキング処理のために送信するイベントを追跡します。通知は、直接、SNS トピックを作成するか、Amazon EventBridge を使用して設定できます。通知の設定の詳細については、「FlexMatchイベント通知の設定 (p. 41)」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。マッチメイ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、マッチメイ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。

リクエストのチケット ID DescribeMatchmakingを指定して呼び出すことで、現在のステータスを含むマッチメイ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、大量のマッチメイ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、マッチメイ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。プレイヤーの承認を管理するプロセスは、Amazon GameLift が管理するホスティングを使用するゲームと、FlexMatchFlexMatchスタンドアロンソリューションとして使用するゲームで同じです。

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) マッチメイ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。このステータスを変更すると、FlexMatch`MatchmakingRequiresAcceptance`イベントがトリガーされます。
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、マッチメイ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。電話で試合の詳細を取得できますDescribeMatchmaking。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. FlexMatch へのプレイヤーの応答のレポート AcceptMatch承諾または拒否のいずれかを指定して電話をかけてプレイヤーの反応を報告します。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、マッチメイキングリクエストが正常に処理されると、ゲームセッション接続情報がマッチメイキングチケットに追加されます。DescribeMatchmaking電話で完了したマッチメイキングチケットを取得する. 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、接続しているプレーヤーをゲームセッションのマッチデータ (チームアサインメントを含む) に関連付けます (を参照 GameSession) 。

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、マッチプレーヤーが適切なゲームセッションに接続できるようにするメカニズムを組み込む必要があります。

マッチメイ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イ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 FlexMatch GameLift がホストするゲームサーバーに追加する

このトピックでは、Amazon FlexMatch 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マッチメイ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。

- GameLiftFlexMatchアマゾンの仕組み (p. 3)
- FlexMatchAmazon GameLift ホスティングとの統合 (p. 8)

このトピックの情報は、「Amazon をゲームサーバーに追加する」で説明されているように、Amazon GameLift Server SDK GameLift がゲームサーバープロジェクトに正常に統合されている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。このトピックのセクションでは、FlexMatch でセットアップしたゲームの処理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 `onStartGameSession()` はゲームセッションオブジェクトを含むリクエストをゲームサーバーに送信します (「」を参照 GameSession)。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 (p. 51)」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イ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。FlexMatch バックフィルの実装の詳細については、「既存のゲームをバックフィルして FlexMatch (p. 52)」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、「既存のゲームをバックフィルして FlexMatch (p. 52)」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オブジェクト内の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。表示されている例では、マッチメイキング設定名は "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}}
   }
  }]
 }]
}
```

既存のゲームをバックフィルして FlexMatch

マッチバックフィルでは、FlexMatch メカニズムを使用して、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マッチメイキングプロセス (p. 4)」を参照してください。

Note

FlexMatchバックフィルは現在、リアルタイムサーバーを使用するゲームでは使用できません。

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を使用すると、1 つ以上のプレーヤースロットが空いて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最初に6人のプレーヤーを見つけてマッチを結成し、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イ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、ゲームサーバーでゲームセッションが開始されるとすぐに、フルではないゲームセッション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 (p. 38)」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) マッチメイ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。マッチメイ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 (p. 13)」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 は、Server SDK GameLift コールバック関数を使用してゲームサーバーをマッチ情報で更新します `onUpdateGameSession` (「サーバープロセスの初期化」を参照)。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 (p. 56)」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、ゲームクライアントまたはゲー

ムサーバーにコードを追加して Amazon GameLift API を呼び出しますStopMatchmaking。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イ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。サーバープロセスには、up-to-dateゲームに接続している現在のプレーヤーと空いているプレイヤースロットのステータスに関するほとんどの情報があります。

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、マッチメイ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、「FlexMatchAmazon GameLift ホスティングとの統合 (p. 8)」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 (p. 56) 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()
   - StopMatchBackfill()

   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、GameLiftゲームセッションをアクティブ化するときに Amazon から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() を呼び出すことで取得できます。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するサーバープロセスには、up-to-date現在どのプレーヤーがゲームセッションに接続しているかの情報が最も多く含まれている必要があります。
  2. プレイヤー ID、属性、チーム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)、MatchmakerDataプロパティ (を参照マッチメーカーデータの操作 (p. 51)) からプレイヤーデータを取得します。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 は、Server SDK GameLift 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ついてゲームサーバーを更新します onUpdateGameSession (「サーバープロセスの初期化」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 (p. 56) 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、StopMatchBackfill() 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、マッチメイ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、「FlexMatchAmazon GameLift ホスティングとの統合 (p. 8)」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 (p. 56)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 Amazon API の AWS SDK を使用します。GameLift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 リファレンスをご覧ください。Amazon GameLift 関連アクションの低レベルのサービス API について説明しており、言語固有のリファレンスガイドへのリンクも含まれています。

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 DescribeGameSessionsを指定して呼び出すことで、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 サービスに接続して、新しいプレーヤー接続を検証したり、プレーヤーがドロップされたことを報告したり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 (p. 46)」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、マッチメイキング設定 ARN のゲームセッションオブジェクト (GameSession)、`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イキングチケット (MatchmakingTicket) には、処理中はゲームセッション ID が付きます。新しいマッチメーキングリクエストのチケットには、マッチが行われるまでゲームセッション ID が割り当てられません。ゲームセッション ID の存在は、新しいマッチのチケットとバックフィルのチケットの違いを見分けるための 1 つの方法です。
- [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() を呼び出すと、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確認できます。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)、MatchmakerDataプロパティ (を参照) マッチメーカーデータの操作 (p. 51) からプレイヤーデータを取得します。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 (マッチメイキングイベントの追跡 (p. 47) 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 (p. 56)」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

ゲーム内でどのようにマッチバックフィルリクエストを開始したとしても、マッチバックフィルリクエストの結果として Amazon GameLift から配信されるゲームセッションの更新をゲームサーバーが処理できなければなりません。

Amazon GameLift がマッチバックフィルリクエストを完了すると (成功するかどうかにかかわらず)、コールバック関数を使用してゲームサーバーを呼び出します。onUpdateGameSessionこの呼び出しには、マッチバックフィルチケット ID、ステータスメッセージ、GameSessionup-to-dateプレイヤー情報を含む

マッチメーキングデータを最も多く含むオブジェクトの 3 つの入力パラメータがあります。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_CANCELED — 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 (サーバー) への呼び出しによってキャンセルされました。StopMatchBackfillGameSession 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. ゲームサーバーが Server SDK action ProcessReady() を呼び出す際に、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。

GameLiftFlexMatchアマゾンリファレンス

このセクションには、Amazonとのマッチメイキングに関するリファレンスドキュメントが含まれています GameLiftFlexMatch.

トピック

- アマゾン GameLift FlexMatch API リファレンス (AWSSDK) (p. 58)
- FlexMatchルール言語 (p. 59)
- FlexMatchマッチメイキングイベント (p. 74)

アマゾン GameLift FlexMatch API リファレンス (AWSSDK)

このトピックでは、Amazon の API オペレーションをタスクベースで一覧表示します。GameLift FlexMatchAmazon GameLift FlexMatch サービス API は、名前空間の AWS SDK にパッケージ化されています。`aws.gamelift`AWSSDK 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ご覧ください。

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、peer-to-peerおよびオンプレミスやクラウドコンピューティングプリミティブなどの他のホスティングシステムでホストされているゲームで使用するためのマッチメイキングサービスを提供しています。その他の Amazon GameLift ホスティングオプションの詳細については、Amazon GameLift 開発者ガイドをご覧ください。

マッチメイ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たり、ゲームのマッチメイキングプロセスを設定したり、マッチやチームを作成するためのカスタムルールセットを定義します。

マッチメイキング設定

- CreateMatchmakingConfiguration— プレイヤーのグループを評価したり、プレイヤーチームを構築したりするための手順を記載したマッチメイキング設定を作成します。Amazon をホスティングに使用する場合は、GameLiftマッチ用の新しいゲームセッションの作成方法も指定してください。
- DescribeMatchmakingConfigurations— Amazon GameLift リージョンで定義されているマッチメイキング設定を取得します。
- UpdateMatchmakingConfiguration—マッチメイキング構成の設定を変更します. キュー.
- DeleteMatchmakingConfiguration— リージョンからマッチメイキング設定を削除します。

マッチメイキングのルールセット

- CreateMatchmakingRuleSet—プレーヤーの試合を検索するときに使用するルール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イキングルールセットを取得します。
- ValidateMatchmakingRuleSet—一連のマッチメイキングルールの構文を確認.
- DeleteMatchmakingRuleSet— リージョンからマッチメイ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イキングリクエストを管理します。

- StartMatchmaking— 同じマッチでプレイしたい1人のプレイヤーまたはグループのマッチメイキングをリクエストします。
- DescribeMatchmaking— ステータスを含め、マッチメイキングリクエストの詳細を取得します。
- AcceptMatch— プレイヤーの承認が必要なマッチについては、GameLiftプレイヤーがマッチ提案を受け入れたらAmazonに通知してください。
- StopMatchmaking— マッチメイキングリクエストをキャンセル.
- StartMatchBackfill-既存のゲームセッションの空いているスロットを埋めるために、追加のプレイヤーマッチをリクエストしてください。

利用可能なプログラミング言語

Amazon をサポートする AWS SDK GameLift は、以下の言語でご利用いただけます。開発環境のサポートについては、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) (アマゾン) GameLift
- Java (SDK ドキュメント) (アマゾン) GameLift
- .NET (SDK ドキュメント) (アマゾン) GameLift
- Go (SDK ドキュメント) (アマゾン) GameLift
- Python (SDK ドキュメント) (アマゾン) GameLift
- Ruby (SDK ドキュメント) (アマゾン) GameLift
- PHP (SDK ドキュメント) (アマゾン) 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 で使用するマッチメイキングルールの作成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。GameLift FlexMatchマッチメイ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、を参照してくださいFlexMatchルールセットを作成 (p. 12)。

トピック

- FlexMatchルールセットスキーマ (p. 60)
- FlexMatchルールセットプロパティ定義 (p. 62)
- FlexMatchルールタイプ (p. 67)
- FlexMatchプロパティ表現 (p. 71)

FlexMatchルールセットスキーマ

FlexMatch ルールセットは、小規模マッチングルールと大規模マッチング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、を参照してくださいFlexMatchルールセットプロパティ定義 (p. 62)。

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。maxPlayersルールセット内のすべてのチームの値の合計が40を超える場合、FlexMatchラージマッチガイドラインに基づいてこのルールセットを使用するマッチ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ルールセットを作成 (p. 12)」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とは関係ありません。この値はマッチが完了したことを表すマッチメイキングデータに含まれますが、Amazon GameLift のプロセスでは使用されません。

許可される値: 文字列

必須? いいえ

`ruleLanguageVersion`

FlexMatch使用されているプロパティ表現言語のバージョン。

許可される値: 1.0

必須? はい

`playerAttributes`

マッチメイキングリクエストに含まれ、マッチメイ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、マッチメイ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イ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? いいえ

`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? はい

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? はい

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? いいえ

algorithm

マッチメイ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? いいえ

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マッチングを行います。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? はい

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? はい

`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アルファ値と数値には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バックフィルチケットをいつバッチで処理するかを決定します。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。
- 「高」— チケットバッチをリクエストタイプ別 (次に年齢別) でソートし、FlexMatch最初にバックフィルチケットの照合を試みます。
- 「low」— チケットバッチはリクエストタイプ別 (次に年齢別) でソートされ、FlexMatchバックフィルされていないチケットを最初に照合しようとします。

必須? いいえ

`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? いいえ

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、マッチメイ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? はい

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? はい

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? はい

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? いいえ

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? いいえ

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? はい

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。マッチメイキングプロセスでは使用されません.

許可される値: 文字列

必須? いいえ

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、を参照してくださいFlexMatch ルールタイプ (p. 67)。

許可される値:

- 「AbsoluteSort」— 指定されたプレイヤー属性がバッチ内の最も古いチケットと比較されるかどうかに基づいて、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してソートします。
- 「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。
- 「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イ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」— 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」— 数値で指定されたプレーヤー属性がバッチ内の最も古いチケットとどのように比較されるかに基づいて、バッチ内のチケットをソートする明示的なソート方法を使用してソートします。
- 「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? はい

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ち時間を計算します。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、マッチメイ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? いいえ

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? はい

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? はい

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
```

バッチ距離ルールは 2 つの属性値の差を測定します。バッチディスタンスルールタイプは、マッチの規模が大きい場合と少ない場合の両方で使用できます。バッチ距離ルールには次の 2 種類があります。

- 数値属性値を比較します。たとえば、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ーヤーが互いに 2 つのスキルレベル以内にいることが求められる場合があります。このタイプでは、batchAttributeすべてのチケット間の最大距離を定義します。
- 文字列の属性値を比較します。たとえば、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、batchAttributeFlexMatchバッチの作成に使用する値を定義します。

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**— マッチが成功するまでの最大距離値。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) FlexMatchのチケットの処理方法を決定する値。有効なオプションには、チケットのプレイヤーの最小値 (max)、最大値 (avg)、平均値 () が含まれます。min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
```

比較ルールは、プレイヤーの属性値を別の値と比較します。比較ルールには次の 2 種類があります。

- 参考値と比較してください。たとえば、このタイプの比較ルールでは、マッチしたプレー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、プレーヤー属性、参照値、および比較演算を指定します。

- プレイヤー間で比較してください。たとえば、このタイプの比較ルールでは、マッチに参加しているすべてのプレーヤーが異なるキャラクターを使用する必要がある場合があります。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

  プレーヤーの属性を比較するには、バッチ距離ルールの方が効率的です。マッチメイキングのレイテンシーを減らすには、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**— 測定値を比較して、一致する見込みがあるかどうかを調べる値。
- **operation**— 測定値を参照値と比較する方法を決定する値。有効な操作には<、、<=、=、!=>、などがあります>=。
- **partyAggregation**— 複数のプレイヤー (パーティ) FlexMatchのチケットの処理方法を決定する値。有効なオプションには、チケットのプレイヤーの最小値 (max)、最大値 (avg)、平均値 () が含まれます。minデフォルトは avg です。

距離ルール

```
distance
```

距離ルールは、プレイヤーのスキルレベル間の距離など、2 つの数値の差を測定します。たとえば、距離ルールでは、すべてのプレーヤーが少なくとも30時間ゲームをプレイしていることが求められる場合があります。

  Note

  プレーヤーの属性を比較するには、バッチ距離ルールの方が効率的です。マッチメイキングのレイテンシーを減らすには、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ーヤー属性値。これは数値の属性でなければなりません。
- **referenceValue**— マッチ候補の距離を測る基準となる数値。
- **minDistance/maxDistance**— マッチが成功した場合の最小距離値または最大距離値。
- **partyAggregation**— 複数のプレイヤー (パーティ) FlexMatchのチケットの処理方法を決定する値。有効なオプションには、チケットのプレイヤーの最小値 (max)、最大値 (avg)、平均値 () が含まれます。min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、プレーヤーの属性値のグループをバッチ内の他のプレー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、複数のプレーヤーの属性値、文字列リストとしてのプレーヤー属性、またはその両方を含めることができます。たとえば、コレクションルールでは、チーム内のプレーヤーが選択したキャラクターを対象とする場合があります。その場合、そのルールでは、チームに特定のキャラクターが少なくとも1人いることが求め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コレクション。属性値は文字列リストでなければなりません。
- **referenceValue**— 測定値を比較して一致するかどうかの判断に使用する値（または値の集まり）。
- **operation**— 一連の測定値を比較する方法を決定する値。有効な操作には以下が含まれます。
  - intersection— このオペレーションは、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。交差演算を使用するルールの例については、を参照してください例 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 (p. 27)。
  - contains— このオペレーションは、指定された参照値を含むプレイヤー属性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、を参照してください例 3: チームレベル要件とレイテンシーの制限を設定する (p. 25)。
  - reference_intersection_count— この操作では、プレーヤー属性コレクション内のアイテムのうち、基準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。この操作を使用して、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、を参照してください例 5: 複数のプレイヤー属性間の交差を見つける (p. 29)。
- **minCount**/**maxCount**— マッチが成功した場合の最小または最大カウント値。
- **partyAggregation**— 複数のプレイヤー（パーティ）FlexMatchのチケットの処理方法を決定する値。unionこの値を使うと、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。

複合規則

```
compound
```

複合ルールでは、40 人以下のプレーヤーで論理的にマッチします。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、一致させ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、基本ルールまたはサポートルールを拡張することはできます (p. 15)。

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口ジック。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。batchDistance複合ルールでは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 つの引数が当てはまる場合、この式は true になります。
  - or— 指定された 2 つの引数のいずれかが当てはまる場合、この式は true になります。
  - not— 式内の引数の結果を逆にします。
  - xor— 引数のうち 1 つだけが当てはまる場合、式は真です。

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、ロケーションごとにプレイヤーのレイテンシーを測定します。レイテンシ
ルールでは、レイテンシが最大値を超えるロケーションは無視されます。レイテンシールールで許可
されるには、プレーヤーのレイテンシー値が少なくとも 1 か所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、一致率が高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。チケットにレイテンシーが最大値を下回るロケーションがない場合、そのチケットはレイテンシールールに一致しません。
- **maxDistance**— 各チケットの待ち時間と距離参照値の間の最大値。
- **distanceReference**— チケットのレイテンシーを比較するレイテンシー値。距離基準値の最大距離内にチケットがあると、マッチングが成功します。有効なオプションには、プレイヤーレイテンシーの最小値 (minavg) と平均 () があります。
- **partyAggregation**— 複数のプレイヤー (パーティ) FlexMatchのチケットの処理方法を決定する値。有効なオプションには、チケットのプレイヤーの最小値 (max)、最大値 (avg)、平均値 () が含まれます。minデフォルトは avg です。

> **Note**
>
> キューを使用すると、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の詳細については、「プレイヤーレイテンシーポリシーの作成」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、バッチに追加された最初の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ソートします。

絶対並べ替えルールのプロパティ

- **sortDirection**—マッチメイキングチケットをソートする順序. 有効なオプションにはascending、およびが含まれますdescending。
- **sortAttribute**— チケットのソート基準となるプレイヤー属性。
- **mapKey**— マップの場合、プレイヤー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— 最も低い値のキーが最初です。
  - maxValue— 最も値の大きいキーが最初です。
- **partyAggregation**— 複数のプレイヤー (パーティ) FlexMatchのチケットの処理方法を決定する値。有効なオプションには、パーティー内のプレーヤーのすべてのプレーヤー属性の最小 (minmax)、最大 (avg) のプレーヤー属性、およびすべてのプレーヤー属性の平均 () が含まれます。デフォルトは avg です。

Example

例

次のルール例では、プレイヤーをスキルレベル別にソートし、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
```

距離ソートルールは、指定されたプレイヤー属性と、バッチに追加された最初のチケットとの距離に基づいて、マッチメーキングチケットのバッチをソートします。

距離並べ替えルールのプロパティ

- **sortDirection**—マッチメイキングチケットをソートする方向. 有効なオプションにはascending、およびが含まれますdescending。
- **sortAttribute**— チケットのソート基準となるプレイヤー属性。
- **mapKey**— マップの場合、プレイヤー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について、最も低い値のキーを探します。
  - maxValue— バッチに最初に追加されたチケットについて、最も値の大きいキーを探します。
- **partyAggregation**— 複数のプレイヤー (パーティ) FlexMatchのチケットの処理方法を決定する値。有効なオプションには、チケットのプレイヤーの最小値 (max)、最大値 (avg)、平均値 () が含まれます。minデフォルトは avg です。

FlexMatchプロパティ表現

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。

一般的なマッチメイキングプロパティ表現

プロパティエクスプレッションは、プレイヤー、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

目標	入力	意味	出力
マッチに参加している特定のチームを特定するには:	`teams[red,blue]`	赤チームと青チーム	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>
マッチ中の特定のチームに属する選手を識別するには:	`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>>
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>>

プロパティエクスプレッションの例

次の表は、前述の例を基にしたいくつかの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、エクスプレッション、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
`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

集約	入力	意味	出力
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>
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	number

式	意味	結果のタイプ
	たリストを取得し、スキルを平均化します。	
count(teams[red].players)	Red チームのプレイヤーの数	number
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マッチメイキングイベント

GameLiftFlexMatchAmazonは、マッチメイキングチケットが処理されるたびにイベントを送信します.「FlexMatchイベント通知の設定 (p. 41)」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。また、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方式で Amazon CloudWatch Events に送信されます。

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示します。MatchmakingTicketマッチメイキングチケットのステータスの詳細については、Amazon GameLift API リファレンスをご覧ください。

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。

資源:ConfigurationArn

詳細:タイプ、チケット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。

資源:ConfigurationArn

詳細:タイプ、チケット、受付タイムアウト、受付必須、、、MatchID ruleEvaluationMetrics gameSessionInfo

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ということは、AcceptMatchそのプレイヤーはまだコールされていないということです。

資源:ConfigurationArn

詳細:タイプ、チケット、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。

資源:ConfigurationArn

詳細:タイプ、チケット、承認、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。

資源:ConfigurationArn

詳細:タイプ、チケット、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。

資源:ConfigurationArn

詳細:タイプ、チケット、メッセージruleEvaluationMetrics、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。

資源:ConfigurationArn

詳細:タイプ、チケット、メッセージruleEvaluationMetrics、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。

資源:ConfigurationArn

詳細:タイプ、チケット、メッセージruleEvaluationMetrics、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。使用時に責任分担モデルを適用する方法についてはFlexMatch、「Amazon のセキュリティ」を参照してくださいGameLift。

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、FlexMatchサービスに関連する新機能、更新、修正に関する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 開発者向けリソース

すべての Amazon GameLift ドキュメントと開発者用リソースを参照するには、Amazon GameLift ドキュメンテーション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。