Amazon GameLift FlexMatch

AWS Black Belt Online Seminar

秋山 周平

Solutions Architect
2023/07

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、 AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は Twitter へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では 2023 年 7 月時点のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます

自己紹介

名前：秋山 周平

所属：アマゾン ウェブ サービス ジャパン合同会社
技術統括本部 Game Tech グループ
エンタープライズソリューション部
ソリューション アーキテクト

経歴：クラウドサポートエンジニア(AWS)
→ ソリューションアーキテクト(AWS)

好きなAWSサービス：


Amazon GameLift


Amazon Cognito


AWS Verified Access

本セミナーの対象者

- マッチメイキングの導入を検討されている方
- 現行のマッチメイキングの仕組みに課題を感じている方
- Amazon GameLift FlexMatch に興味を持たれている方

アジェンダ

1. ゲームにおけるマッチメイキング
2. Amazon GameLift FlexMatch の概要
3. FlexMatch 使用開始の手順
4. マッチメイキング設定
5. マッチメイキングルール
6. ステータス一覧と遷移
7. 料金情報
8. まとめ

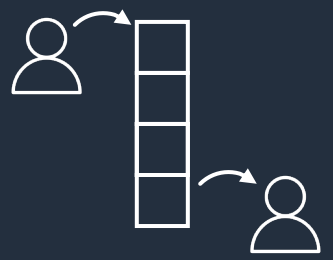
ゲームにおける
マッチメイキング

マッチメイキングとは

不特定多数のプレイヤーが接続するオンラインのゲームにおいて、
プレイヤーの集合から2人以上のグループを生成する手続き

特徴

プレイヤーではなく
サービスが組み合わせ
(マッチ)を決定

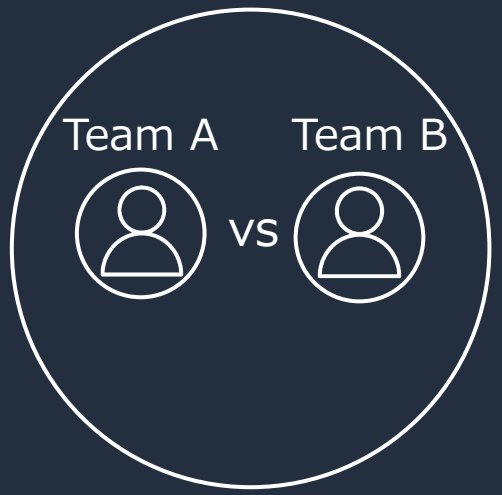


動的に人が出入りする
プレイヤーの集合

組み合わせを決める
様々な条件

マッチメイキングのユースケース例

オンライン対戦
格闘ゲーム



2つの陣営
1陣営1人

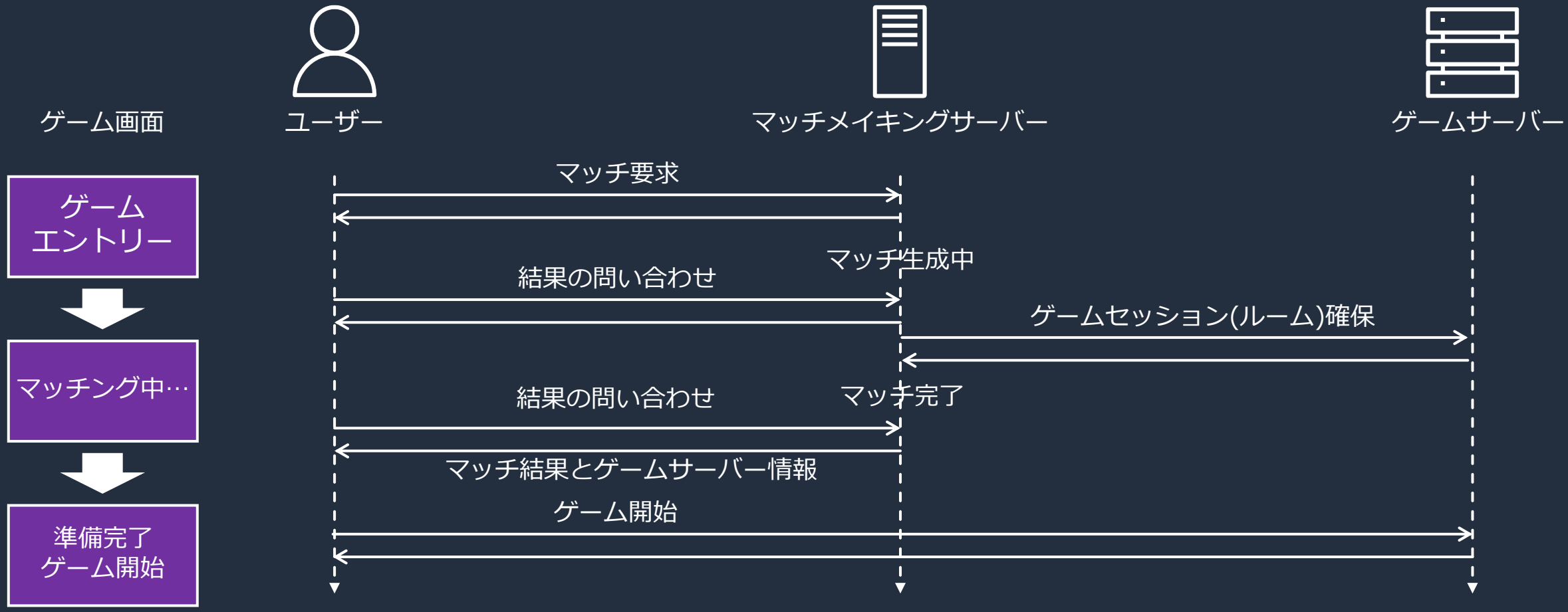
バトルロワイヤル形式
オンラインゲーム

7つの陣営
1陣営３人チーム

オンラインの
パーティーゲーム

1つの陣営
３～５人のチーム

シーケンス図でみるマッチメイキング(ポーリングの例)



マッチメイキング機能実現の困難さ

ゲーム特有のアルゴリズム開発

- 組み合わせ爆発への対応
- プレイヤーの動的な入退出

頻繁に発生するロジックの修正

- パラメータのチューニング
- ゲームの仕様変更に伴うロジック変更

多様な要件への対応

- マッチ後のプレイヤー離脱
- マッチ後のプレイヤーの追加加入

インターフェースの実装

- API の実装
- ログ、メトリクスの出力

Amazon GameLift FlexMatch 概要

Amazon GameLift FlexMatch 概要

フルマネージドで柔軟なカスタマイズが可能なマッチメイキングサービスを提供

AWS によるフルマネージドな
マッチメイキングエンジン

ロジックやマッチの構造を
管理が容易なJSONで記述

柔軟かつ豊富な
マッチメイキングロジック

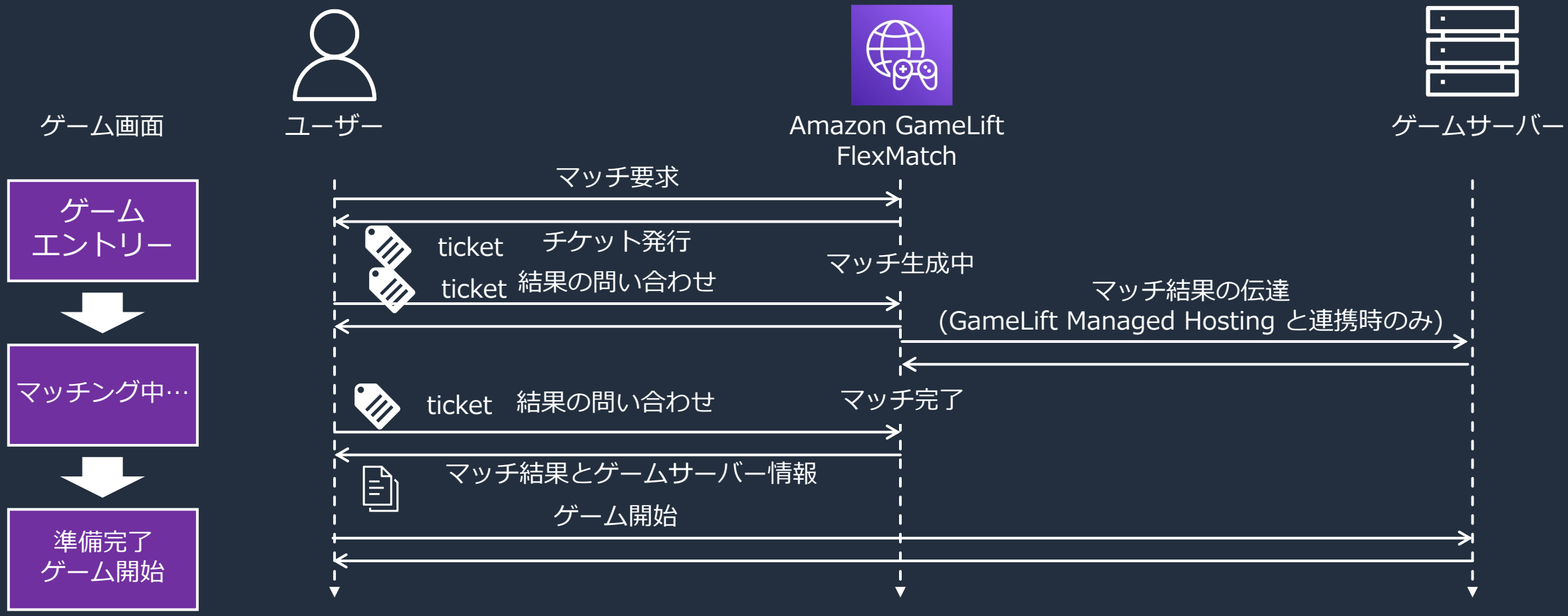
最大 200 人のマッチ

AWSサービスとの連携

- CloudWatch へメトリクスの出力
- SNS へのイベント通知
- CloudTrail で操作ログ記録

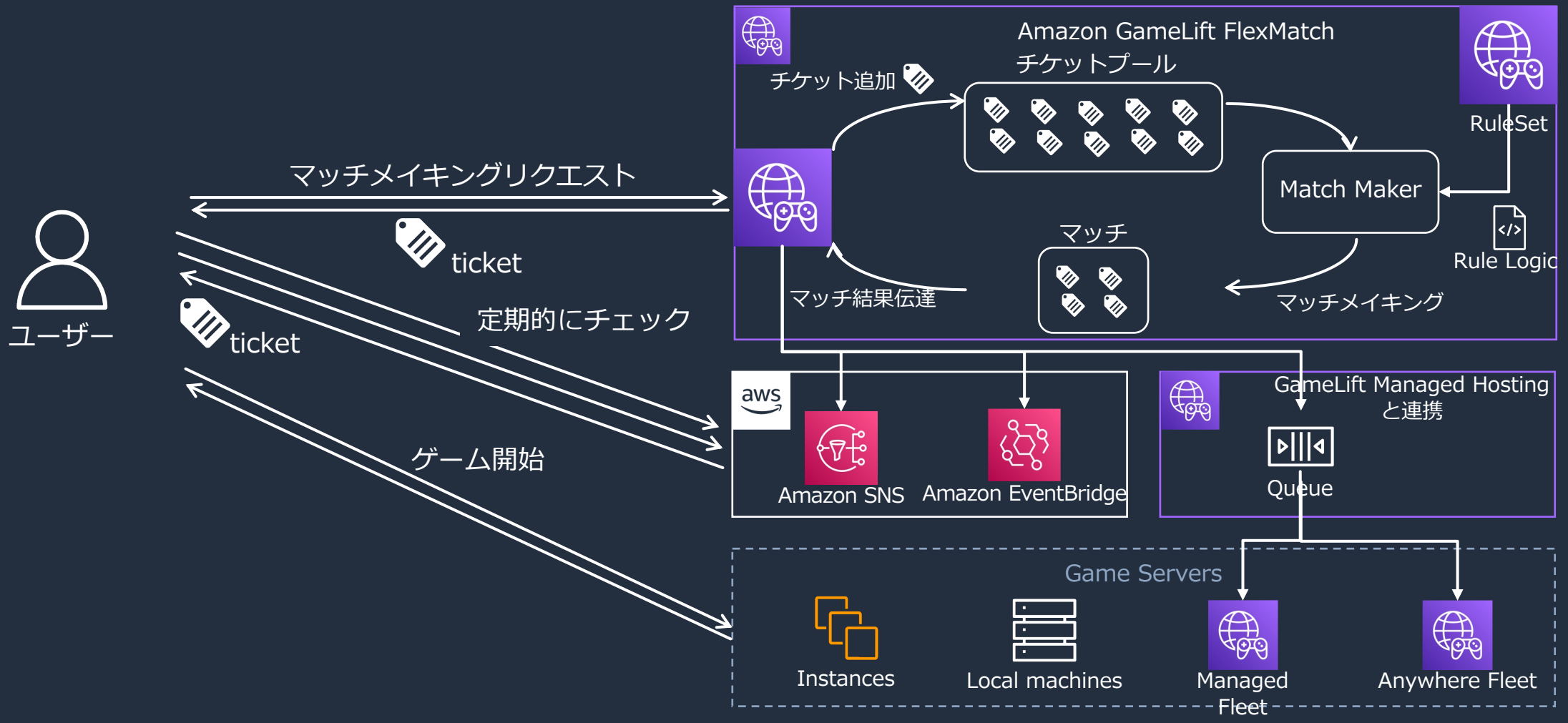
シーケンス図でみるマッチメイキング(ポーリングの例)

マッチメイキングサーバーとして FlexMatch を利用



Amazon GameLift FlexMatch の全体像

FlexMatchはチケット発行、マッチ生成、他サービスへの通知を行う



Amazon GameLift FlexMatch コアコンポーネント

FlexMatch に存在する2つのコアリソース

マッチメイキング設定（マッチメーカー）

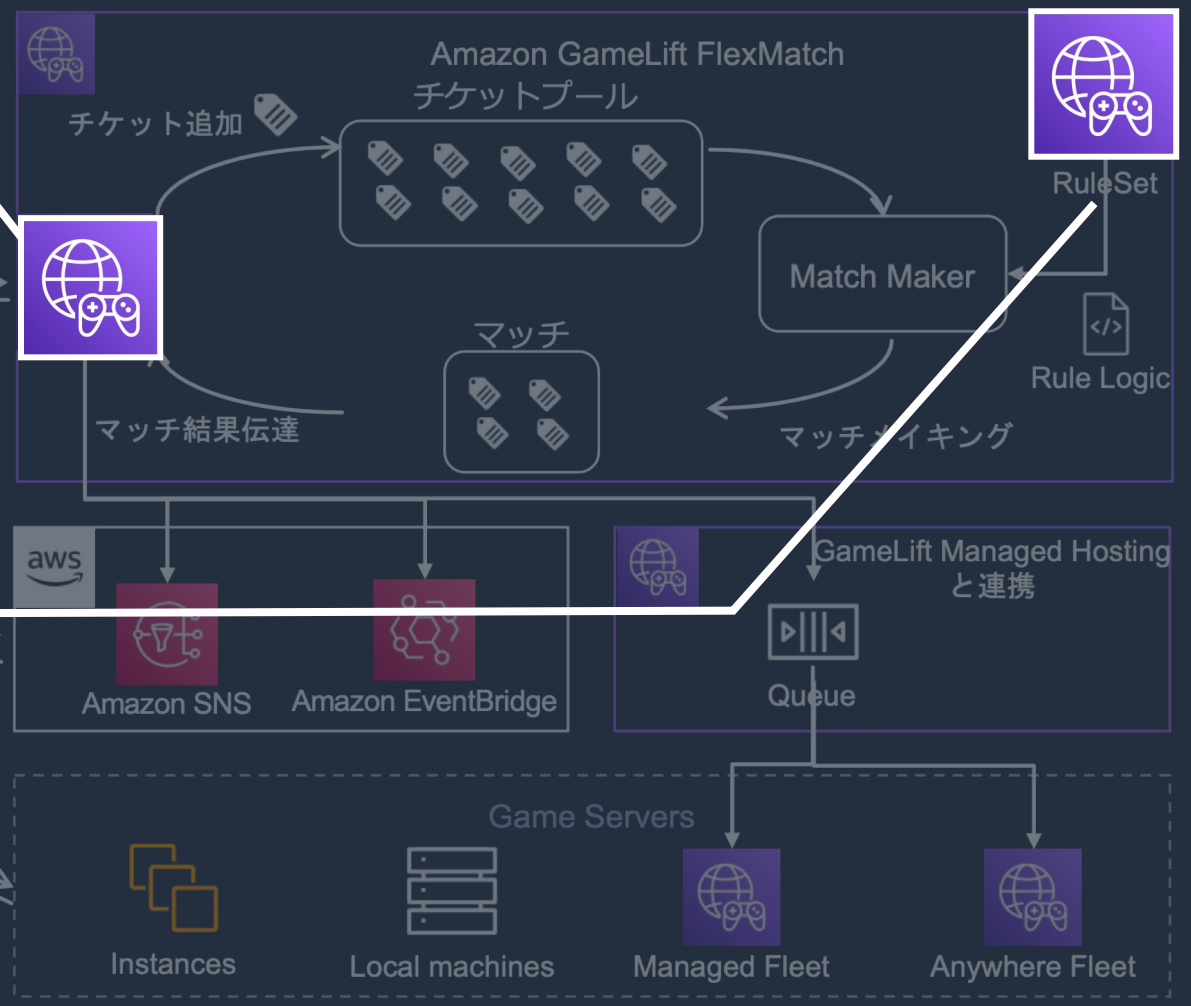
マッチメイキングを実行するリソース

- プレイヤーはマッチメイキング設定を指定してリクエスト
- 1つのルールセットを持つ

ルールセット

マッチメイキングのルールを定義するリソース

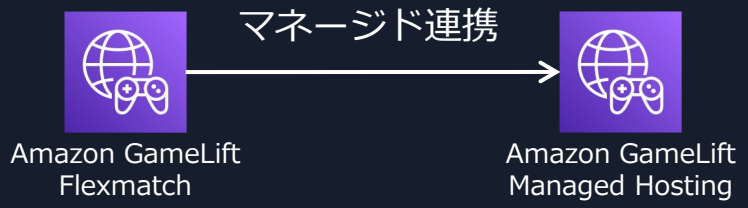
- ゲームのチーム構造とサイズ
- マッチするプレイヤーの条件



FlexMatch と他 GameLift サービスの連携

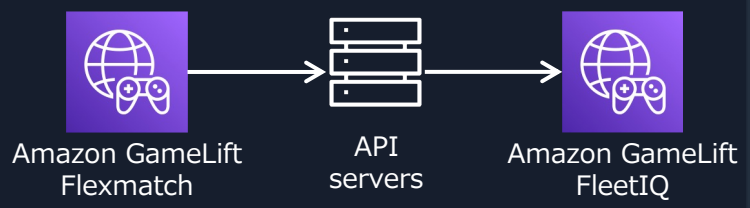
FlexMatch は Managed Hosting とのみマネージドの連携機能が利用可能

Amazon GameLift Managed Hosting と連携



FlexMatch の機能で連携

Amazon GameLift FleetIQ と併用

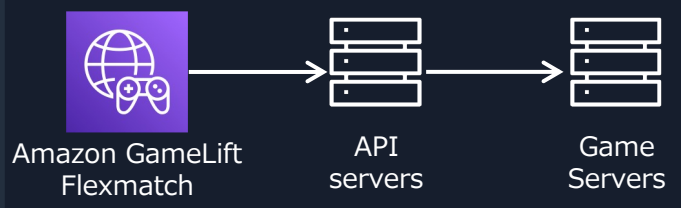


FlexMatch との連携には独自の仕組みが必要

マッチ結果の取得・管理

サーバーへの結果の送信

独自のゲームサーバーと併用



FlexMatch との連携には独自の仕組みが必要

マッチ結果の取得・管理

サーバーへの結果の送信

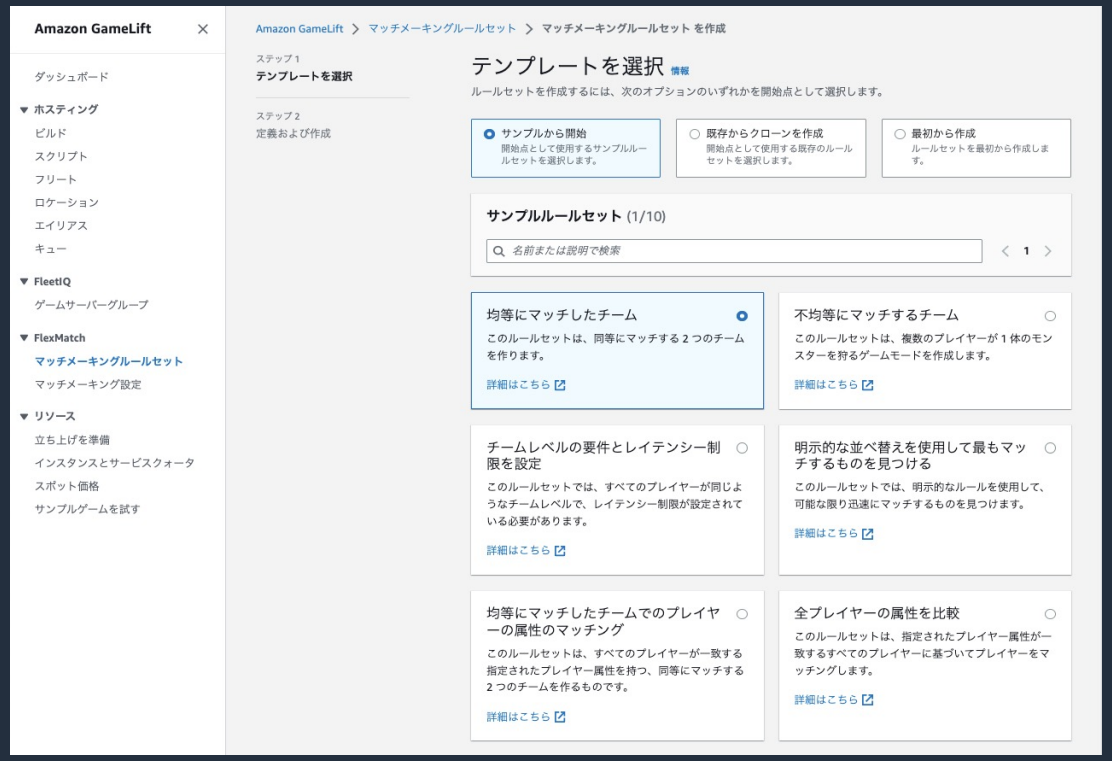
FlexMatch 使用開始の手順

FlexMatch 使用開始の手順

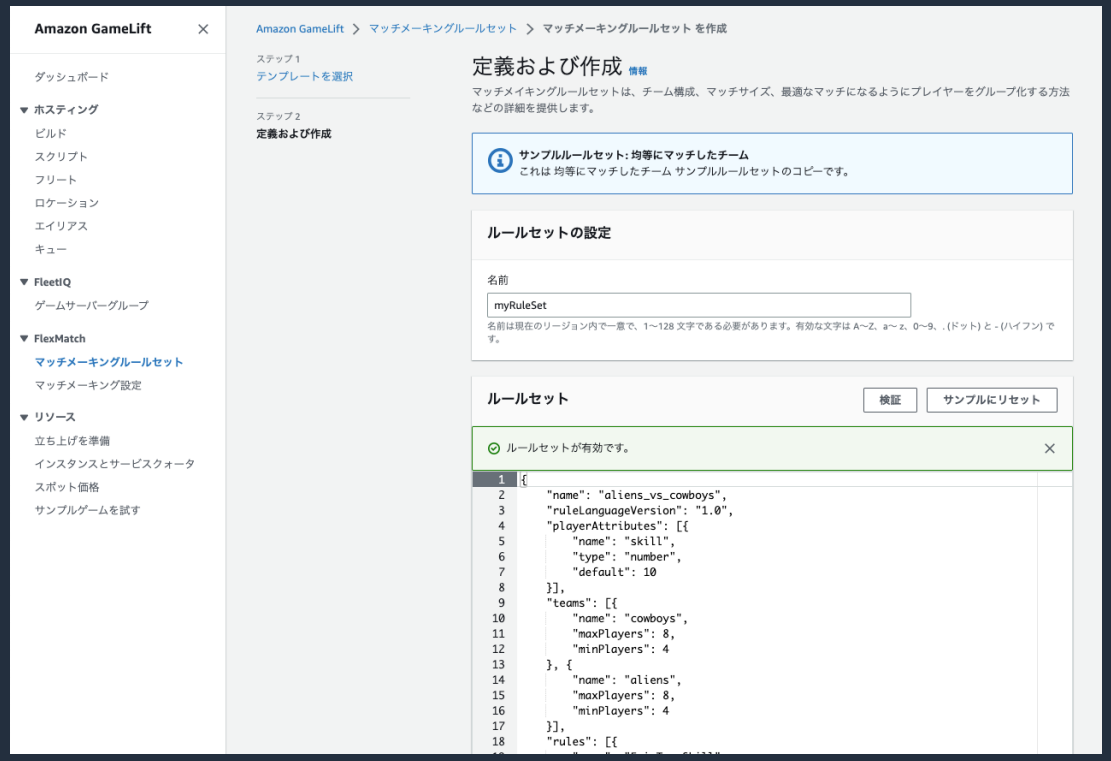
4 stepでマッチメイキングを開始

Step1 マッチメイキングルールセットの作成

AWS コンソールでサンプルルールセットを選択



ルールセットの名前「myRuleSet」を入力し作成

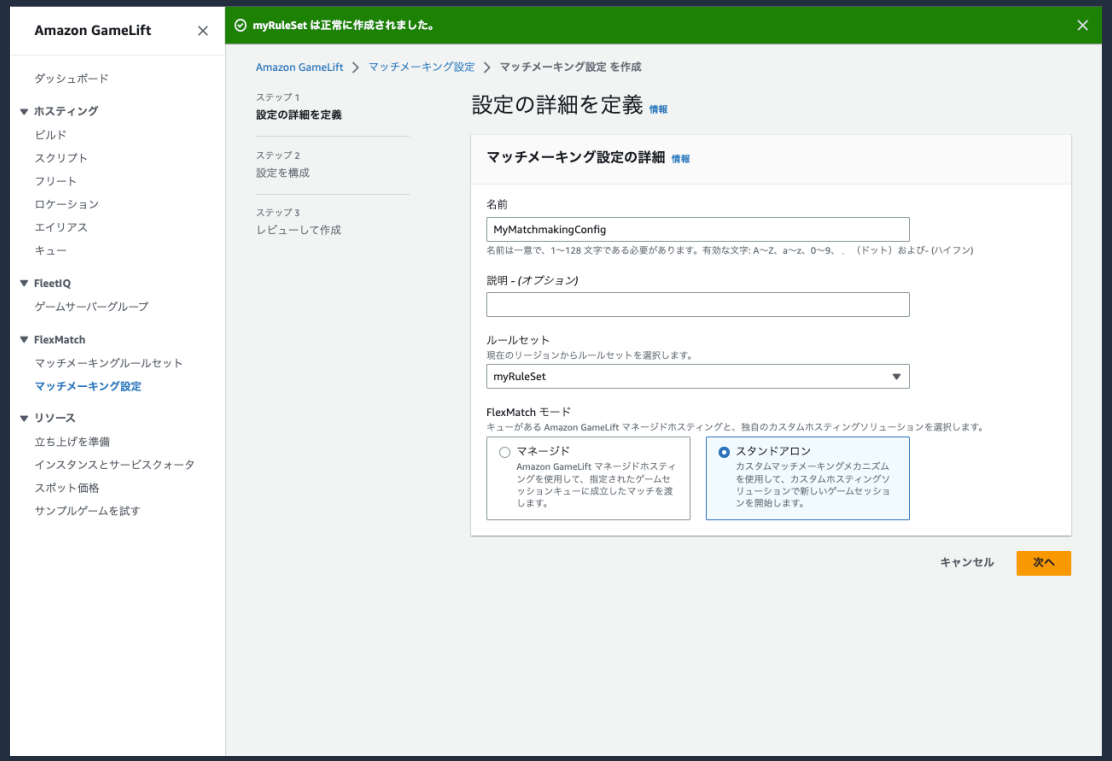


FlexMatch 使用開始の手順

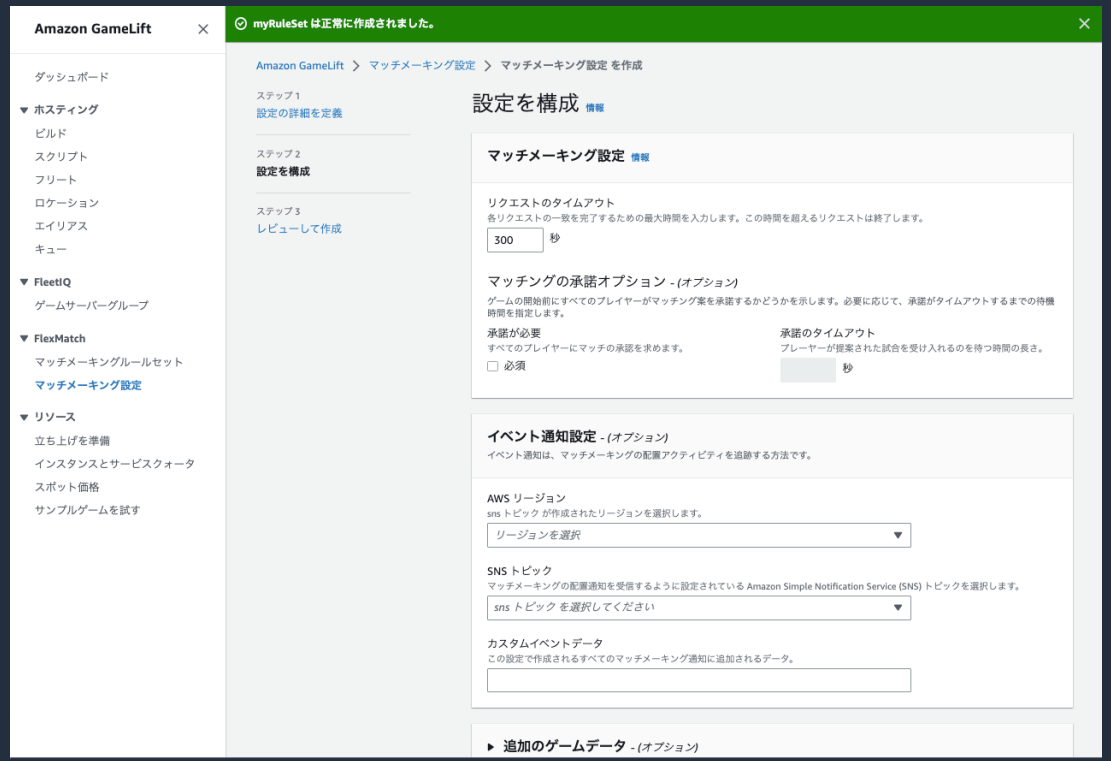
4 stepでマッチメイキングを開始

Step2 マッチメイキング設定の作成

「MyMatchmakingConfig」という名前を選択後、
Step1で作成したマッチメイキングルールセットを選択
さらに、「スタンドアロン」を選択



リクエストのタイムアウトを「300秒」に設定し
そのほかはデフォルトのまま作成



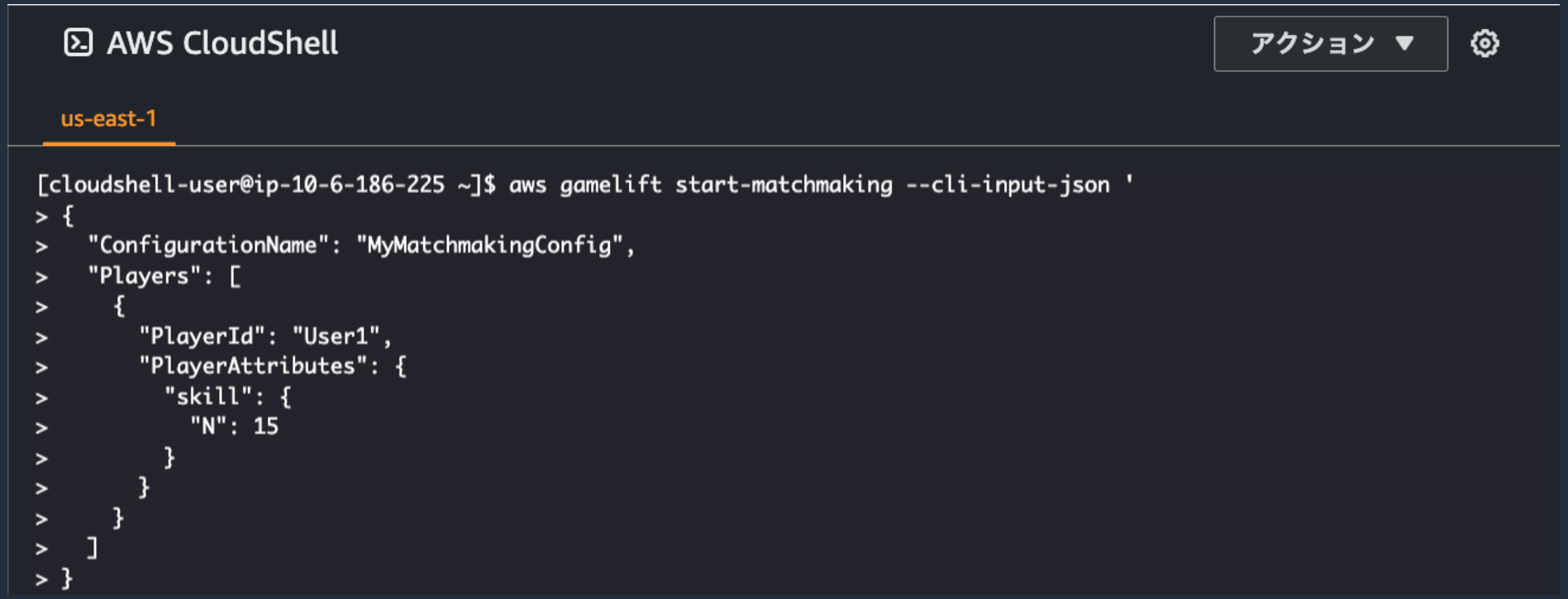
FlexMatch 使用開始の手順

4 stepでマッチメイキングを開始

Step3 マッチメイキングリクエストの開始

AWS CLI が実行できる環境にて、StartMatchmaking API を実行

今回のマッチは最小 8 人必要なため、300秒以内に 8 人分の StartMatchmaking API を実行



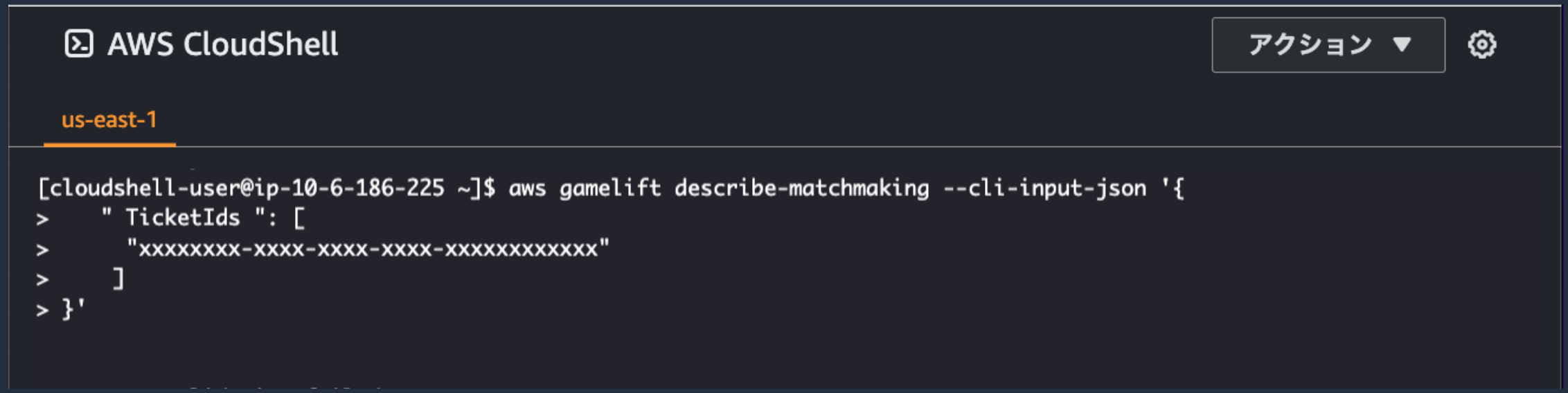
FlexMatch 使用開始の手順

4 stepでマッチメイキングを開始

Step4 マッチメイキングリクエストの結果の確認

StartMatchmaking の結果で取得した チケットID を使用し DescribeMatchmaking API を実行

発行したチケットのステータスが確認可能（マッチ完了を示す COMPLETED やタイムアウトを示す TIMED_OUT 等)



マッチメイキング設定

マッチメイキング設定

マッチメイキングの動作設定や入出力などロジック以外の部分を設定

通知

- FlexMatch モード
- マッチメイキングのステータス通知

マッチメイキングの動作

- バックフィル
- マッチ承認
- タイムアウト

その他

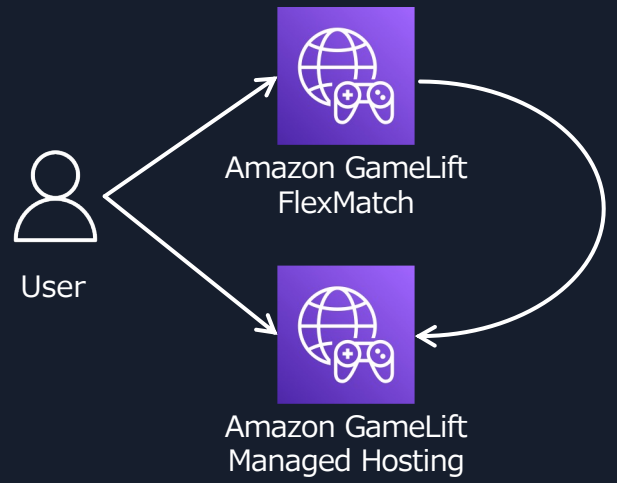
- マッチメイキング設定名
- カスタムデータ
- イベントデータ
- タグ

FlexMatch モード

GameLift Managed Hosting との連携

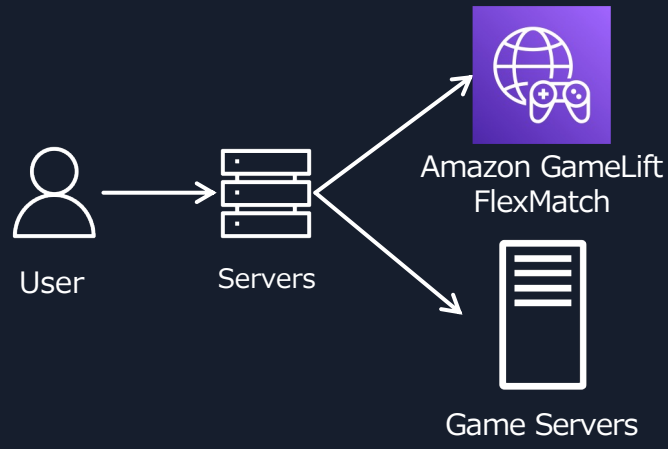
WITH_QUEUE (連携)

FlexMatch でマッチ成立後、自動でゲームセッションを作成



STANDALONE (単独利用)

マッチの結果を取得しゲームサーバーに伝える独自の仕組みが必要



FlexMatch モード

WITH_QUEUE と STANDALONE モードの差異

WITH_QUEUE (連携)

- GameLift Managed Hosting との完全な連携
- GameLift キューによるセッションの自動配置
- Automatic バックフィル機能のサポート
- FlexMatch 利用コストが無料

STANDALONE (単独利用)

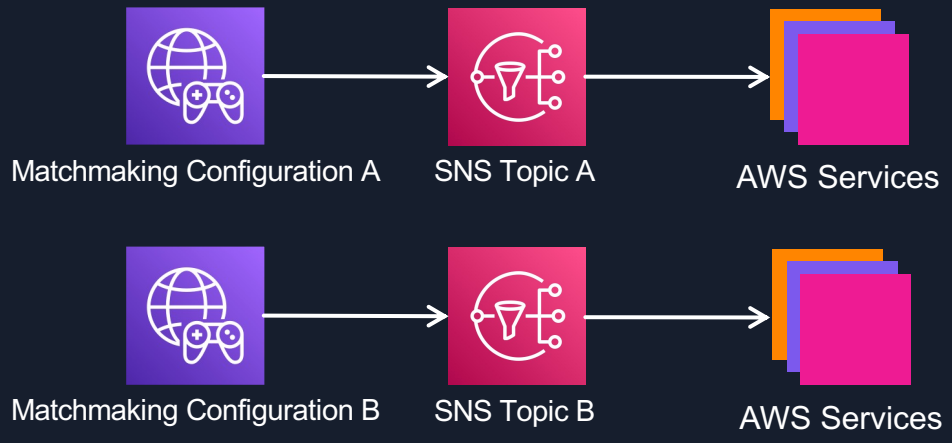
- あらゆるゲームやサービスと連携
- ゲームセッションの配置とマッチの連携は実装が必要
- 独立した利用コストが発生

マッチメイキングイベントのステータス通知

SNS Topic と EventBridge でイベントを受信

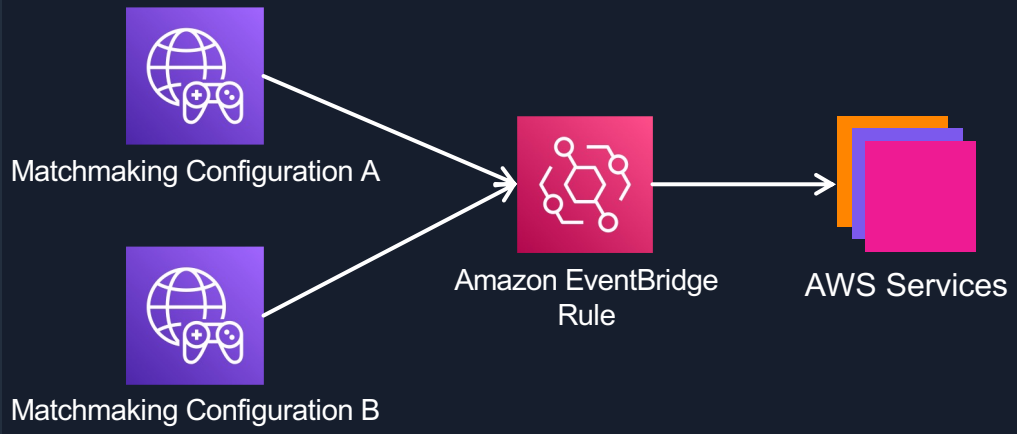
SNS Topic

マッチメイキング設定ごとに
宛先を指定



EventBridge

全マッチメイキング設定のイベント
を受け取るため EventBridge 側に
フィルタが必要



個々のマッチメイキングのステータス通知

イベント結果のクライアントへの通知

GameLift へ直接結果を確認するポーリングは現時点では非推奨
以下のようにイベントを別ストレージに保存することで対応

バックフィル

マッチ成立し進行中のゲームを終了せず新規のプレイヤーを追加する機能

ユースケースに応じた2種類のバックフィルに対応

BackfillMode：AUTOMATIC

最大人数より少ない人数で開始し、プレイヤー枠に空きがある状態で進行中のゲームに、GameLift が自動でプレイヤーを補充

BackfillMode：MANUAL

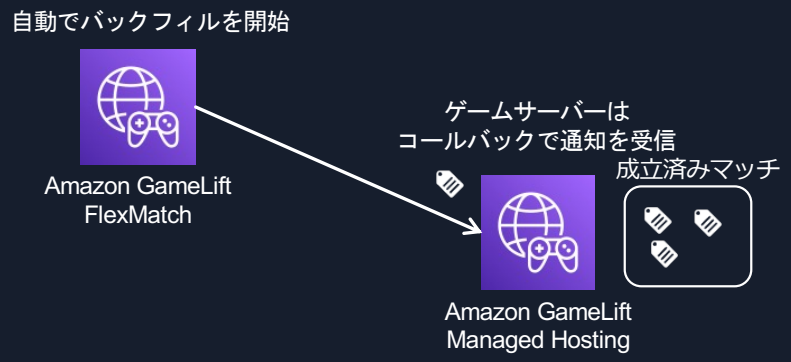
プレイヤーがドロップした時などに、ゲームサーバーが新たなユーザー補充を GameLift に要求

マッチメイキングイベントのステータス通知

FlexMatch モードごとにバックフィルの挙動が変化

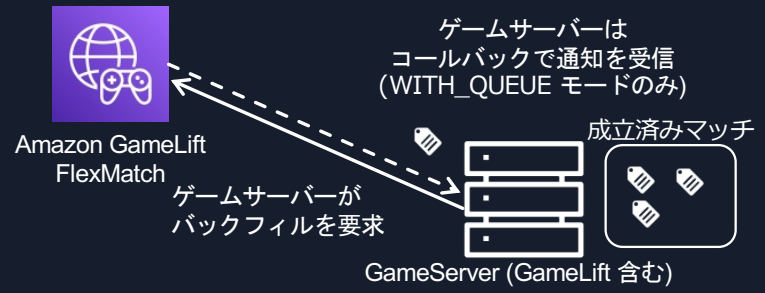
BackfillMode：AUTOMATIC

- WITH_QUEUE モードのみ設定可
- FlexMatch が自動でバックフィルを実行し、GameLift Managed Hosting のゲームサーバーに通知
- ゲームサーバー側はコールバックでバックフィルの結果を受信



BackfillMode：MANUAL

- FlexMatch へのバックフィルリクエストを起点にバックフィルを開始
- WITH_QUEUE モードのみゲームサーバー側はコールバックで結果を受信
- STANDALONE モードでは結果を問い合わせる必要がある

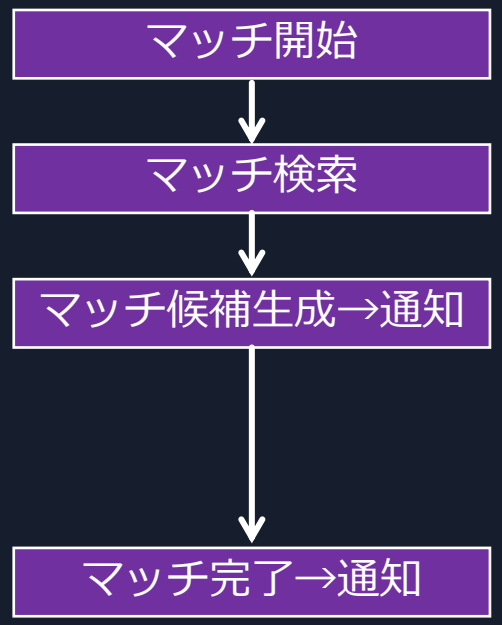


マッチ承認オプション

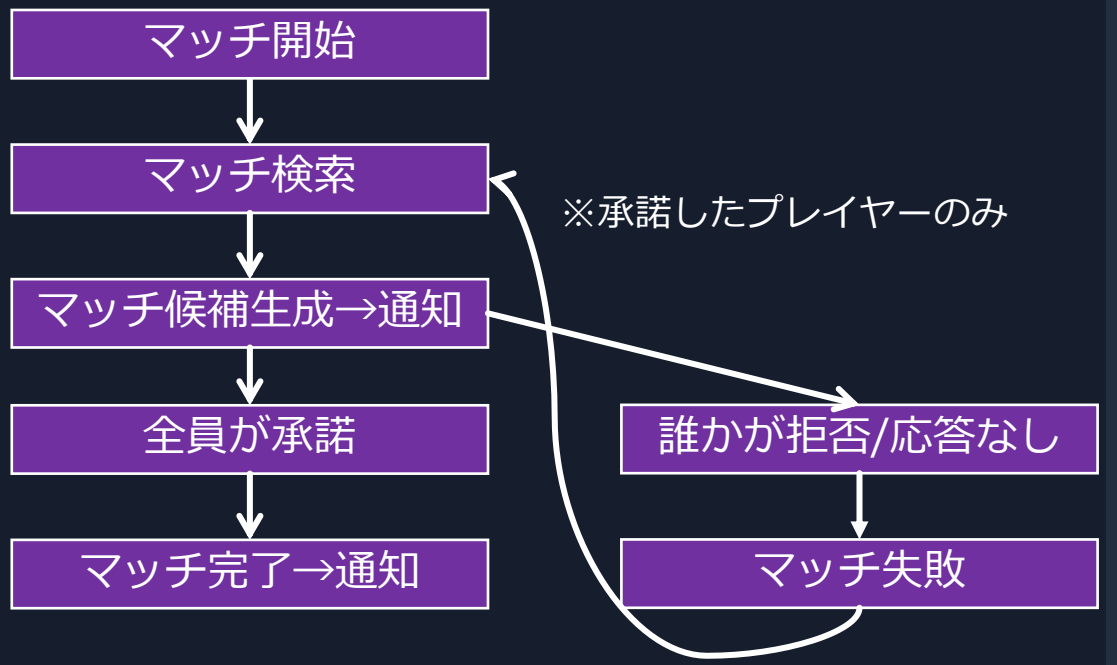
マッチメイキングプロセスに、プレイヤーからの承認を追加

承認オプションの設定によるフローの差異

Not Required (デフォルト)



Required



タイムアウト設定

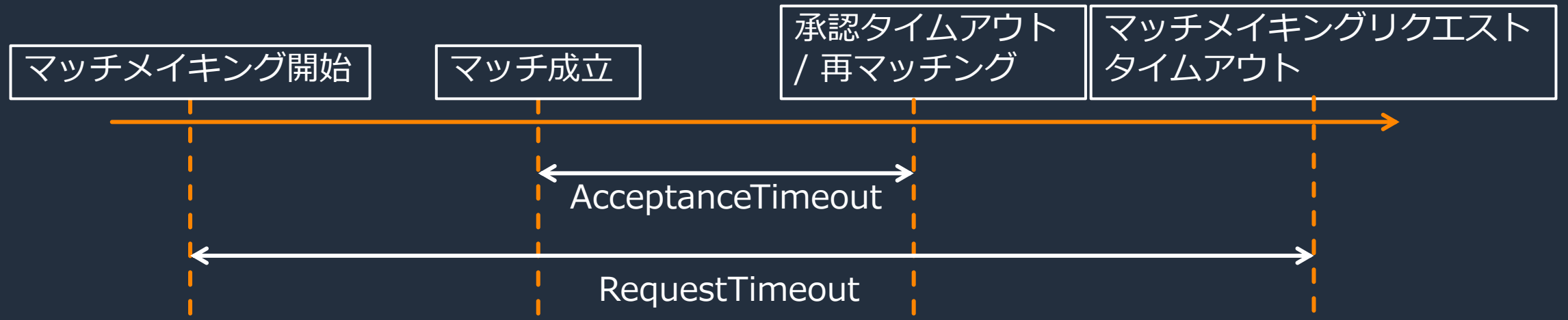
2種類のタイムアウトが存在

RequestTimeout

発行したチケットが有効な時間(秒)

AcceptanceTimeout

マッチ成立後、ユーザーからの承認を待つ時間(秒)



その他の設定

マッチメイキング設定名
- CloudWatch メトリクスに発行するデータのディメンションに使用
- StartMatchmaking API でチケットを発行する際に名前で指定

ゲームプロパティ
- GameLift Managed Hosting と連携時、 ゲームサーバーが値を受け取り可能
- Key-Value 形式のデータ

カスタムイベントデータ
- 単一の文字列
- マッチメイキングに関するイベント通知に付与

タグ
- 複数リソースの管理、分類

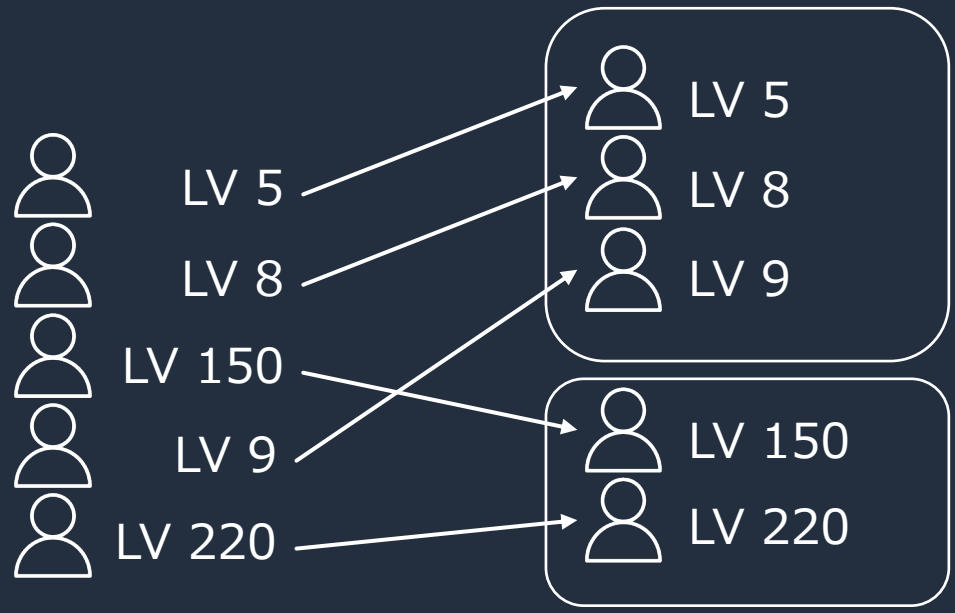
ルールセット

ルールセット概要

マッチメイキングの時のユーザー条件や構造を JSON で記述

特徴

- 多様なチーム構造
- ユーザー属性に応じた
  マッチメイキング
- 時間経過による条件の緩和
- 少人数向け（スモールマッチ）と
  大人数向け（ラージマッチ）の
  2 種類のモード



ユーザー属性に応じた
マッチメイキングの例

ルールセットの JSON プロパティ

トップレベルのプロパティ一覧

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{...}],
    "algorithm": {...},
    "teams": [{...}],
    "rules": [{...}],
    "expansions": [{...}]
}
```

必須のプロパティ
- ruleLanguageVersion
- teams

他の要素と連携するプロパティ
- expansions
  - rules もしくは teams に影響
- rules
  - playerAttributes を参照
- algorithm
  - 設定によって一部ルール使用不可

ルールセット：teams

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{...}],
    "algorithm": {...},
    "teams": [{...}],
    "rules": [{...}],
    "expansions": [{...}]
}
```

複数のチームを記載することでマッチにおけるチームの対立関係を定義

- maxPlayers と minPlayers が異なる場合、状況に応じて異なる人数でマッチが成立
- 同じチーム構造の場合、quantity を使用することで記述を省略可能
- チームの人数を非対称にすることで、非対称な関係も表現可能

ルールセット：teams 記載例

```
"teams": [{
        "name": "team_red",
        "maxPlayers": 3,
        "minPlayers": 3,
        "quantity": 2
    },{
        "name": "team_blue",
        "maxPlayers": 1,
        "minPlayers": 1,
        "quantity": 1
    }],
```

以下チーム構造のマッチを実現

ルールセット：playerAttributes

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{...}],
    "algorithm": {...},
    "teams": [{...}],
    "rules": [{...}],
    "expansions": [{...}]
}
```

マッチ生成時に参照するユーザー属性

- StartMatchmaking API で属性を FlexMatch に渡す
- default 属性でデフォルト値を指定

型	対応する値の例
string	"value"
number	0.0
string_list	["a","b"]
string_number_map	{"a":1,"b":2}

ルールセット：playerAttributes 記載例

```
"playerAttributes": [{
        "name": "rank",
        "type": "number",
        "default": 0
    },{
        "name": "gameMode",
        "type": "string",
        "default": "FreeMatch"
}],
```

後述の rules で参照可能な属性を定義

Rank	gameMode
2	"FreeMatch"

Rank	gameMode
10	"RankMatch"

パラメータを参照したルールの例

- rank が近いユーザー同士のマッチ
- 事前定義されたプレイ形式から同じ形式を選んだユーザー同士のマッチ

ルールセット：expansions

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{...}],
    "algorithm": {...},
    "teams": [{...}],
    "rules": [{...}],
    "expansions": [{...}]
}
```

時間経過により rules もしくは teams で指定した値を変更

経過時間と上書きする値を指定

対象フィールド

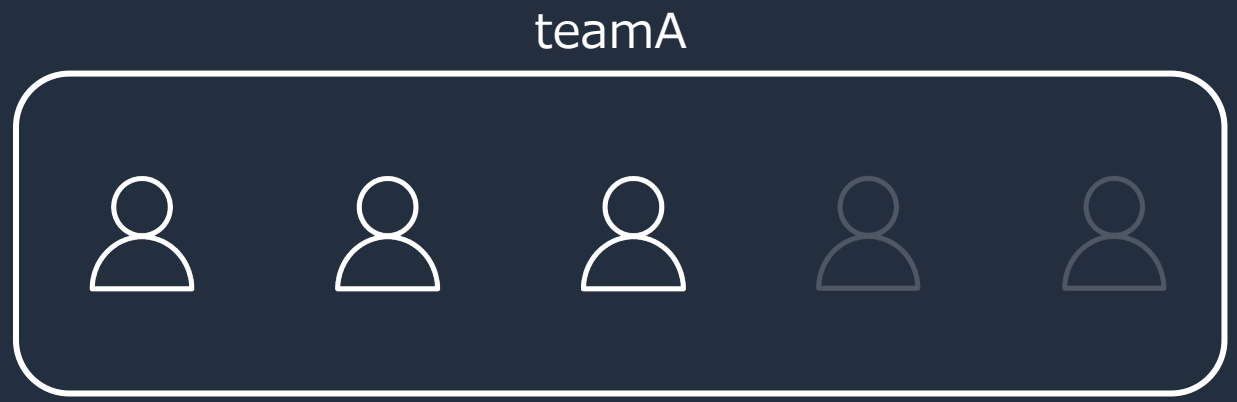
- rules のプロパティ
- teams のサイズプロパティ

ルールセット：expansions 記載例

```
 "teams": [{
        "name": "teamA",
        "maxPlayers": 5,
        "minPlayers": 5
    }],
"expansions": [{
        "target": "teams[teamA].minPlayers",
        "steps": [{
            "waitTimeSeconds": 60,
            "value": 3
        }]
    }]
```

60秒経過時点で teamA の最小人数を5人から3人に変更

60秒経過時点で5人のマッチが成立しない場合、3人でマッチを成立とする



ルールセット：algorithm

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{...}],
    "algorithm": {...},
    "teams": [{...}],
    "rules": [{...}],
    "expansions": [{...}]
}
```

マッチメーカーの挙動を変更する設定

プロパティ名	説明
strategy	ラージマッチもしくはスモールマッチを指定
batchingPreference	チケットプールの事前ソート方法を指定
balancedAttribute	ラージマッチの事前ソートで使用するプレイヤー属性を指定
sortByAttributes	スモールマッチの事前ソートで使用するプレイヤー属性
backfillPriority	マッチにおけるバックフィルチケットの優先度
expansionAgeSelection	expansions における経過時間の基準を指定

ラージマッチとスモールマッチ

41人以上のプレイヤーのマッチを扱う場合 algorithm で指定が必要

スモールマッチ

maxPlayers が 40人以下

スモールマッチで指定可能な algorithm の設定

```
"algorithm": {
      "strategy": "exhaustiveSearch",
      "batchingPreference": <"random", "sorted">,
      "sortByAttributes": [ "string", "string" ],
      "expansionAgeSelection": <"newest", "oldest">,
      "backfillPriority": <"normal", "low", "high">
   },
```

ラージマッチ

maxPlayers が41人以上

ラージマッチで指定可能な algorithm の設定

```
 "algorithm": {
       "strategy": "balanced",
       "batchingPreference": <"largestPopulation",
"fastestRegion">,
       "balancedAttribute": "string",
       "expansionAgeSelection": <"newest", "oldest">,
       "backfillPriority": <"normal", "low", "high">
    },
```

ルールセット：rules

```
{
    "name": "string",
    "ruleLanguageVersion": "1.0",
    "playerAttributes":[{...}],
    "algorithm": {...},
    "teams": [{...}],
    "rules": [{...}],
    "expansions": [{...}]
}
```

マッチを生成する時の条件を指定

type で多様なルールを定義可能

- batchDistance
- comparison
- distance
- collection
- compound
- latency
- absoluteSort
- distanceSort

- 定義したルールは原則上から順番に評価

rules：パラメータの評価方法

多くのルールでは比較用の 2 種類のプロパティを使用

- measurements：プレイヤー（チーム）ごとの属性値（のリスト）
- referenceValue：比較に使用する値（のリスト）

StartMatchmaking および StartMatchBackfill API で一度に複数のプレイヤーを含めた場合、同一グループとなる

ルール上での評価は、グループの平均の値となるが以下のプロパティで変更可能

- partyAggregation：min / max / avg

rules：評価式の記述方法

比較のための値の算出は集約関数とプロパティ表現を使用

```
"rules": [{
        "type": "distance",
        "name": "levelcheck",
        "measurements":
["avg(teams[teama].players.attributes[LEVEL])"],
        "referenceValue":
"avg(teams[teamb].players.attributes[LEVEL])",
        "maxDistance": 10
    }
]
```

集約関数

- リスト中の要素を集計し単一の値を算出することで次元を削減

プロパティ表現

- ルールで参照可能な値の指定方法

rules：プロパティ表現

プロパティ表現と値の記述例

```
{
   "teams": [{
      "red": {
         "players": [{
               "playerId":"player1",
               "attributes": [{"level": 5}]
            },{
               "playerId":"player2",
               "attributes": [{"level": 10}]
         }]
      },
      "blue": {
         "players": [{
               "playerId":"player3",
               "attributes": [{"level": 12}]
            },{
               "playerId":"player4",
               "attributes": [{"level": 1}]
         }]
      }
   }]
}
```

表現例	型	値
teams[red].players.attributes[level]	List<number>	[5,10]
teams[red, blue].players.attributes[level]	List<List<number>>	[[5,10], [12,1]]
teams[*].players.attributes[level]	List<List<number>>	[[5,10], [12,1]]
teams[red].players[playerId]	List<string>	["player1", "player2"]

rules：集約関数

集約関数と値の記述例

```
{
   "teams": [{
      "red": {
         "players": [{
               "playerId":"player1",
               "attributes": [{"level": 5}]
            },{
               "playerId":"player2",
               "attributes": [{"level": 10}]
         }]
      },
      "blue": {
         "players": [{
               "playerId":"player3",
               "attributes": [{"level": 12}]
            },{
               "playerId":"player4",
               "attributes": [{"level": 1}]
         }]
      }
   }]
}
```

表現例	型	値
avg(teams[*].players.attributes[level])	List<number>	[7.5,6.5]
flatten(teams[*].players.attributes[level])	List<number>	[5,10,12,1]
avg(teams[red].players.attributes[level])	number	7.5

Rules：ルールタイプ

比較に用いるルール

- batchDistance
- comparison
- distance
- collection
- latency
- compound

マッチ生成時のプレイヤー検索順序に影響を与えるルール

- absoluteSort
- distanceSort

rules：batchDistance ルール

プレイヤー属性名を指定し全プレイヤーを相互比較
属性値の型が数値の場合は属性値間の距離による比較が可能

例：StageName 文字列が一致するか評価

```
{
    "name": "SimilarSkillRatings",
    "type": "batchDistance",
    "batchAttribute": "StageName"
}
```

例： Skill 数値の相互距離が 500 以下か評価

```
{
    "name": "SimilarSkillRatings",
    "type": "batchDistance",
    "batchAttribute": "Skill",
    "maxDistance": 500
}
```

rules：comparison ルール

2通りの比較が可能
- プレイヤーごとに referenceValue と measurements を比較
- 全プレイヤー間で measurements が 「等しい」か「等しくない」かを評価

例：チーム red のプレイヤー数が4であるか評価

```
{
    "name": "EqualTeamSizes",
    "type": "comparison",
    "measurements":
["count(teams[red].players)"],
    "referenceValue": "4",
    "operation": "="
}
```

例：全プレイヤーの mode 属性が一致するか評価

```
{
    "name": "SameModeComparison",
    "type": "comparison",
    "measurements":
["flatten(teams[*].players.attributes[mode]
)"] ,
    "operation": "="
}
```

rules：distance ルール

measurements で指定した値のリストと referenceValue 特定の値を比較

例：チームごとの number 属性 skill の平均が全プレイヤーの skill 属性の平均から距離 500 以内か評価

```
{
   "name": "SimilarSkillRatings",
   "type": "distance",
   "measurements": ["avg(teams[*].players.attributes[skill])"],
   "referenceValue": "avg(flatten(teams[*].players.attributes[skill]))",
   "maxDistance": 500
}
```

rules：collection ルール

measurements で指定した文字列のリストと複数の方法で比較

例：チームごとに、チーム中のプレイヤーの
string 属性 mode の重複が 1 以下であるか評価

```
{
  "name":"SimilarSkillRatings",
  "type":"collection",
  "operation":"intersection",
  "measurements" :
["teams[*].players.attributes[mode]"],
  "minCount": 1
}
```

比較方法	説明
intersecion	measurements 中の重複する値の数を評価
contains	measurements 中の referenceValue で指定した値と一致する数を評価
reference_intersection_count	measurements の各要素ごとに referenceValue のリストの要素のうち一致する数を評価

rules：compound ルール

他のルールを指定し、指定したルール同士を評価
評価元のルールの単独の結果はマッチ全体の成否の評価から除外

例： RuleA と RuleB のどちらかが ture か評価

```
{"name":"RuleA",…},
{"name":"RuleB",…},
{
    "name": "CompoundRuleExample",
    "type": "compound",
    "statement": "or(RuleA,RuleB)"
}
```

使用可能な 比較演算子

- and
- or
- not
- xor

rules：latency ルール

プレイヤーが StartMatchmaking で報告したレイテンシーを評価

例：プレイヤーが報告したレイテンシー情報のうち、
最大が 100 以下のリージョンが存在するか評価

```
{
    "name": "FastConnection",
    "type": "latency",
    "maxLatency": 100
}
```

プレイヤーはリージョンごとの
レイテンシーを報告

評価に合致するリージョンが1つ以上
存在するかどうかを評価

rules：distanceSort / absoluteSort ルール

チケットを事前にソートすることで、ソートされた属性の評価を高速化

例：事前にチケットを mapPreference 属性の最大値で降順にソート

```
{
    "name": "MapPreference",
    "type": "absoluteSort",
    "sortDirection": "descending",
    "sortAttribute": "mapPreference",
    "mapKey": "maxValue"
}
```

ルールごとにソート基準が異なる

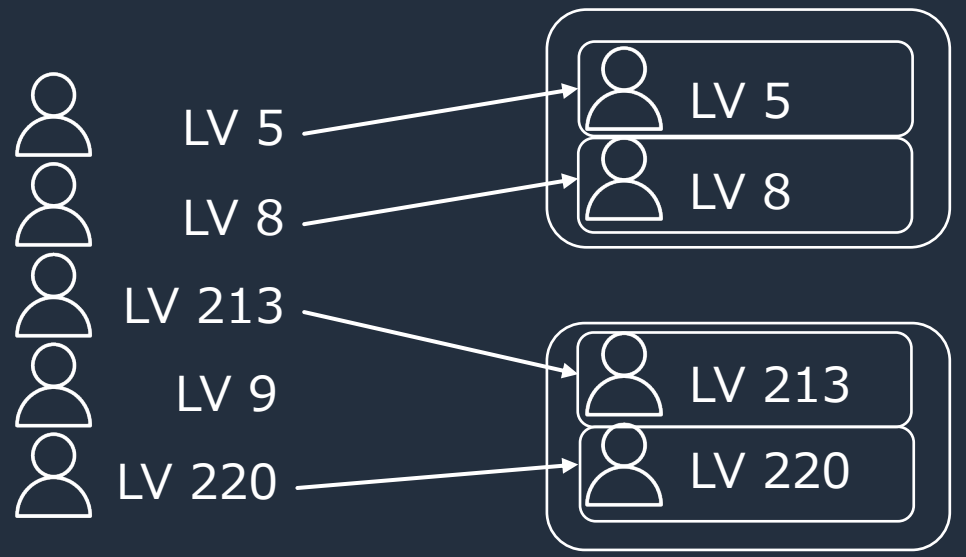
distanceSort	基準となる特定チケットの属性からの距離でソート
absoluteSort	指定した属性値の大きさによるソート

string_number_map を使う場合、map 中のどの数値を参照値とするか mapkey で指定

ルールの例 1

```
{
   "name": "rankmatch",
   "ruleLanguageVersion": "1.0",
   "playerAttributes": [{
      "name": "LEVEL",
      "type": "number"
   }],
   "teams": [{
      "name": "teama",
      "maxPlayers": 1,
      "minPlayers": 1
   },{
      "name": "teamb",
      "maxPlayers": 1,
      "minPlayers": 1
   }],
    "rules": [{
       "type": "distance",
       "name": "level",
       "measurements": ["avg(teams[teama].players.attributes[LEVEL])"],
       "referenceValue": "avg(teams[teamb].players.attributes[LEVEL])",
       "maxDistance": 10
    }]
}
```

レベル差が10以内となるような 2 人のマッチを生成

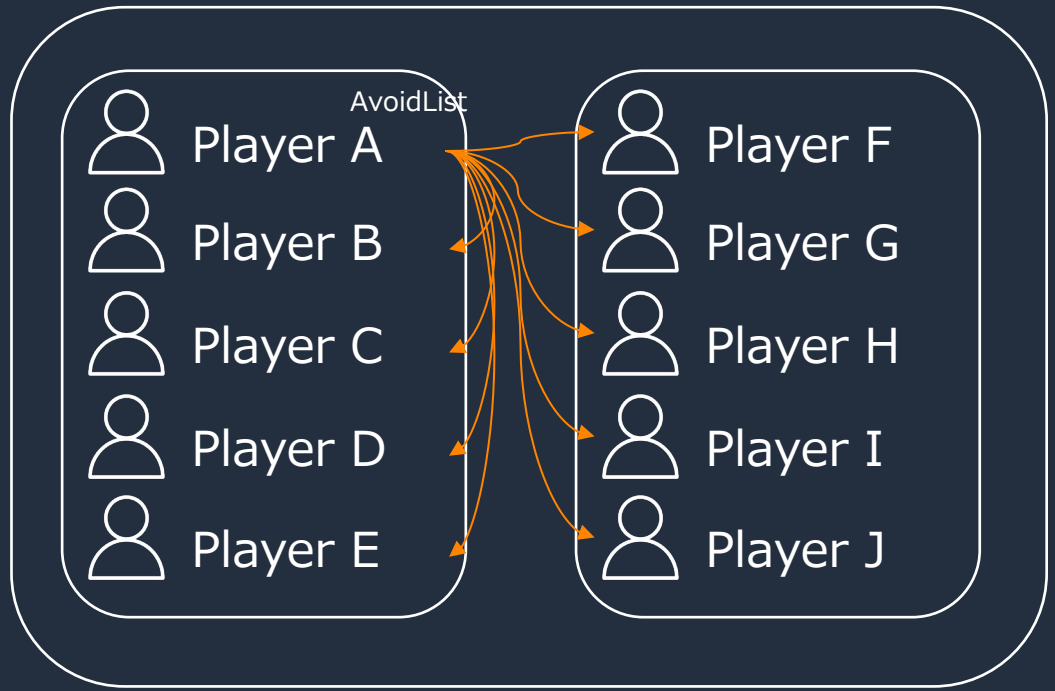


ルールの例 2

```
{
    "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"
    }],
    "rules": [{
        "name": "PlayerIdNotInAvoidList",
        "type": "collection",
        "operation": "reference_intersection_count",
        "measurements": ["flatten(teams[*].players.attributes[AvoidList])"],
        "referenceValue": "flatten(teams[*].players[playerId])",
        "maxCount": 0
    }]
}
```

互いに過去に対戦したことがない
プレイヤー 10 人のマッチを生成

Player A の AvoidList 比較時の図



FlexMatch におけるステータスと遷移

マッチメイキング設定と複雑度

ステータスの遷移や通知されるイベントの量は FlexMatch モードと承認オプションによって変化

- 承認オプション：承認プロセスが全体のフローに追加
- FlexMatchMode：マッチ生成後のゲームセッション配置プロセスが全体のフローに追加

複雑度：大

複雑度：小

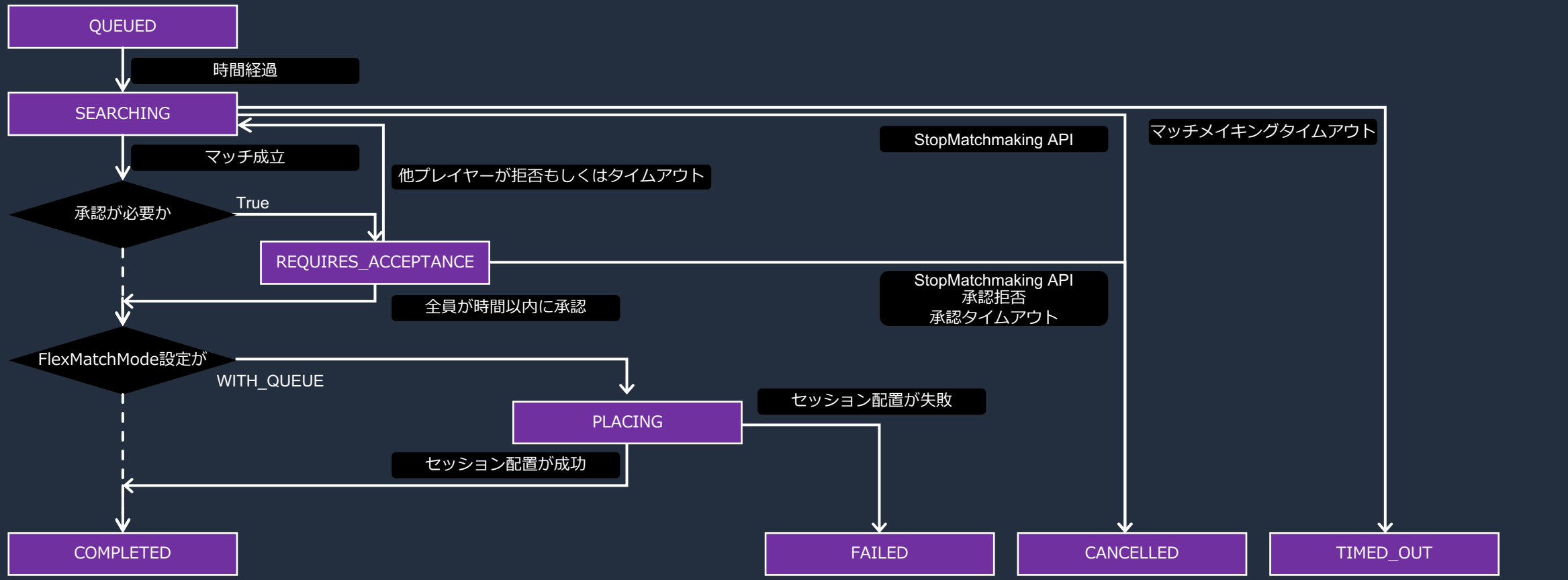
• 承認オプション 承認あり • FlexMatchMode WITH_QUEUE	• 承認オプション 承認なし • FlexMatchMode WITH_QUEUE	• 承認オプション 承認あり • FlexMatchMode STANDALONE	• 承認オプション 承認なし • FlexMatchMode STANDALONE

チケットのステータス

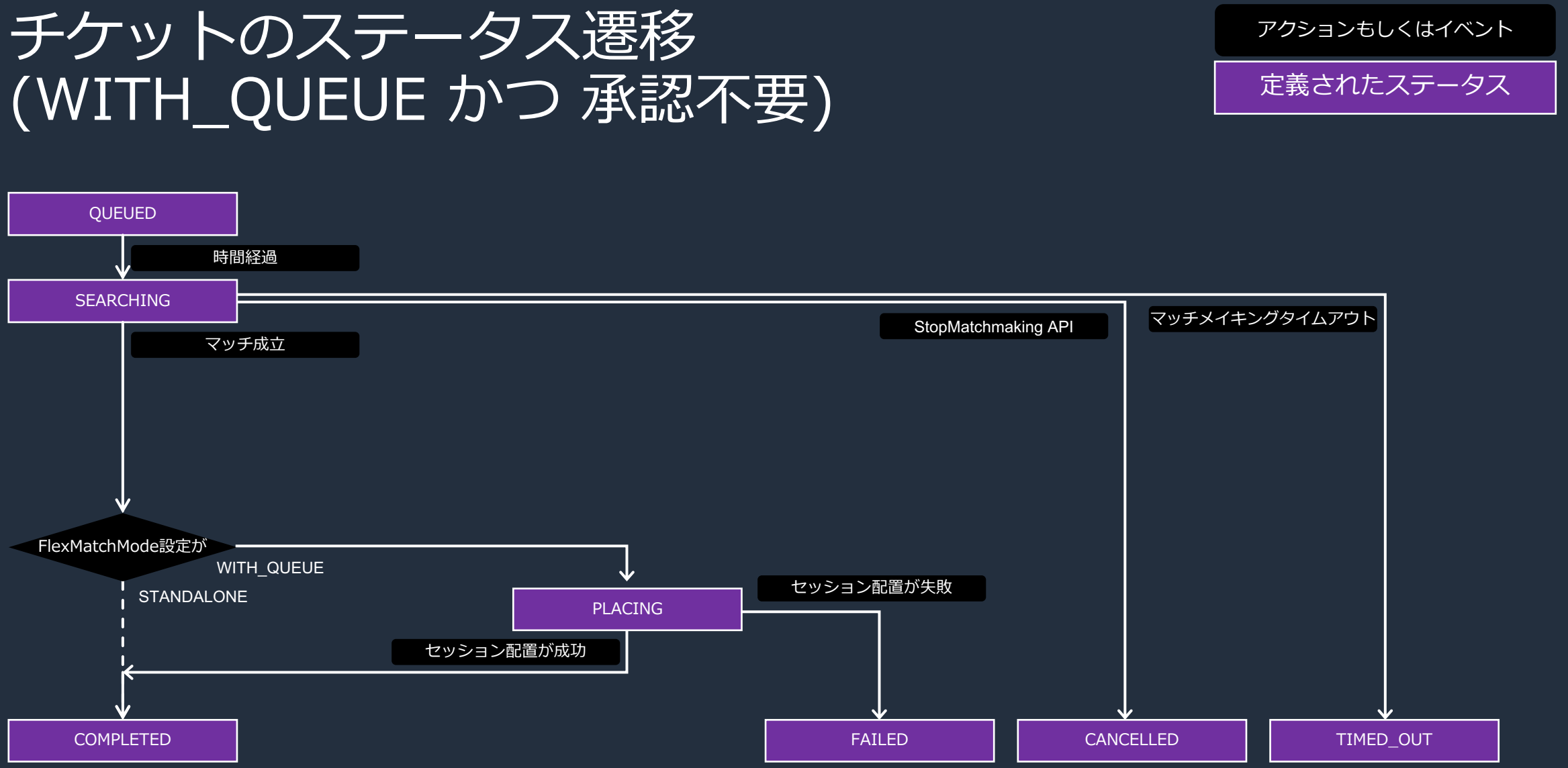
マッチメイキングリクエストのチケットは状況に応じてステータスが変化

- QUEUED : リクエストは受け付けられたが、マッチメーカーで処理される前
- SEARCHING : マッチメーカーで処理中
- REQUIRES_ACCEPTANCE : マッチは成立したが、承認待ちの状態
- PLACING : マッチが成立したが、WITH_QUEUE モードのため GameSession を作成中
- COMPLETED : マッチが成立し、ゲームセッションの配置も完了
- FAILED : 何らかの理由で処理が失敗した
- CANCELLED : StopMatchmaking API で中断した場合 / AcceptMacth API で承認を拒否
- TIMED_OUT : 時間以内にマッチが成立しなかった

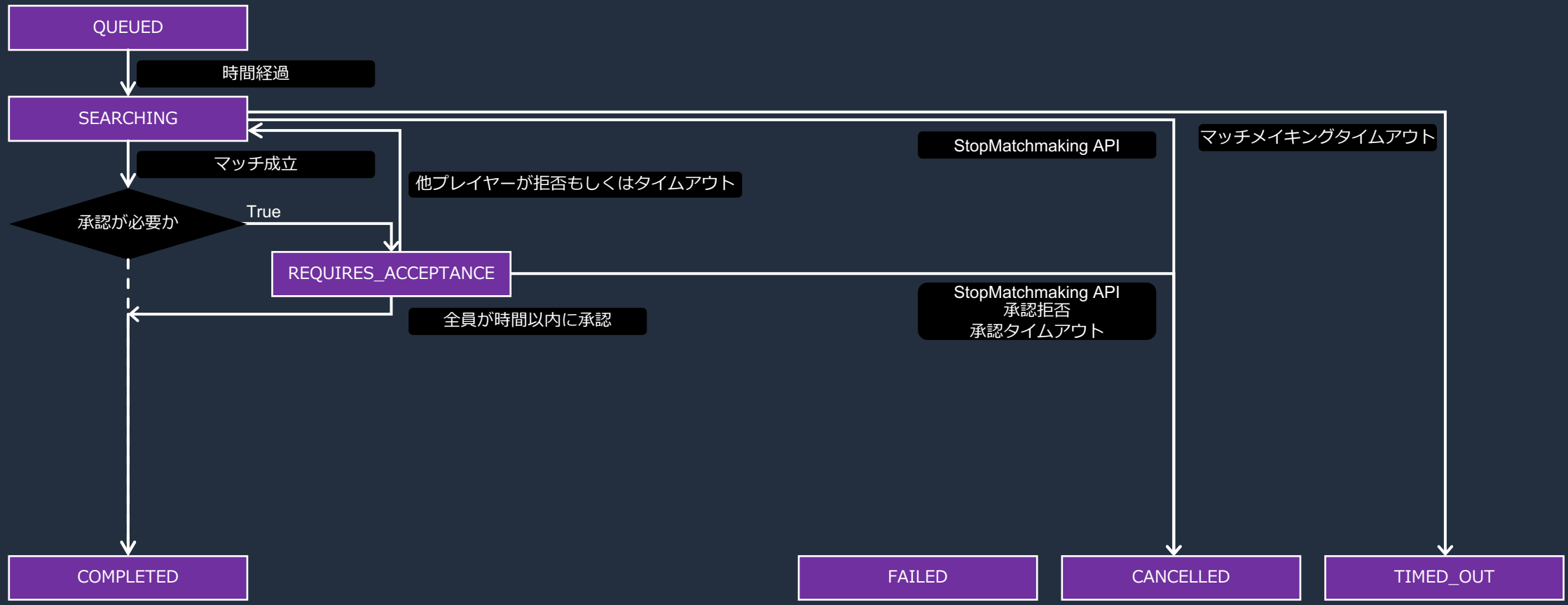
チケットのステータス遷移全体図 (WITH_QUEUE かつ 承認必要)



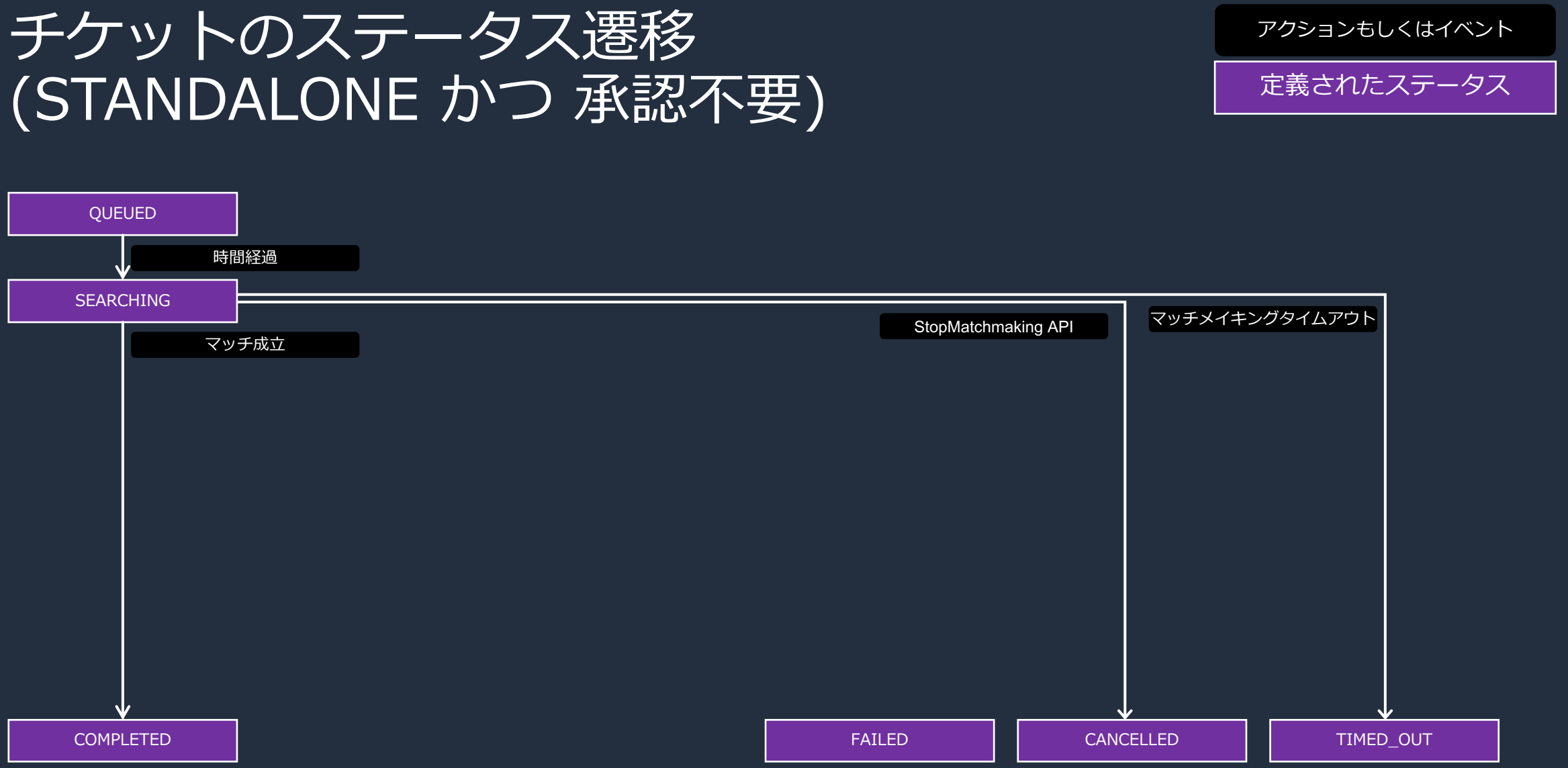
チケットのステータス遷移 (WITH_QUEUE かつ 承認不要)



チケットのステータス遷移 (STANDALONE かつ 承認必要)



チケットのステータス遷移 (STANDALONE かつ 承認不要)



通知ステータス

マッチとチケットの状況に応じて SNS Topic と EventBridge へ通知

個別のチケットに関する通知

- MatchmakingSearching　　：チケットがマッチング処理対象になった時に出力
- AcceptMatch　　：マッチ承諾時に出力
- MatchmakingCancelled　　：キャンセルした場合に出力
- MatchmakingTimedOut　　：タイムアウトした場合に出力

マッチ全体に関する通知

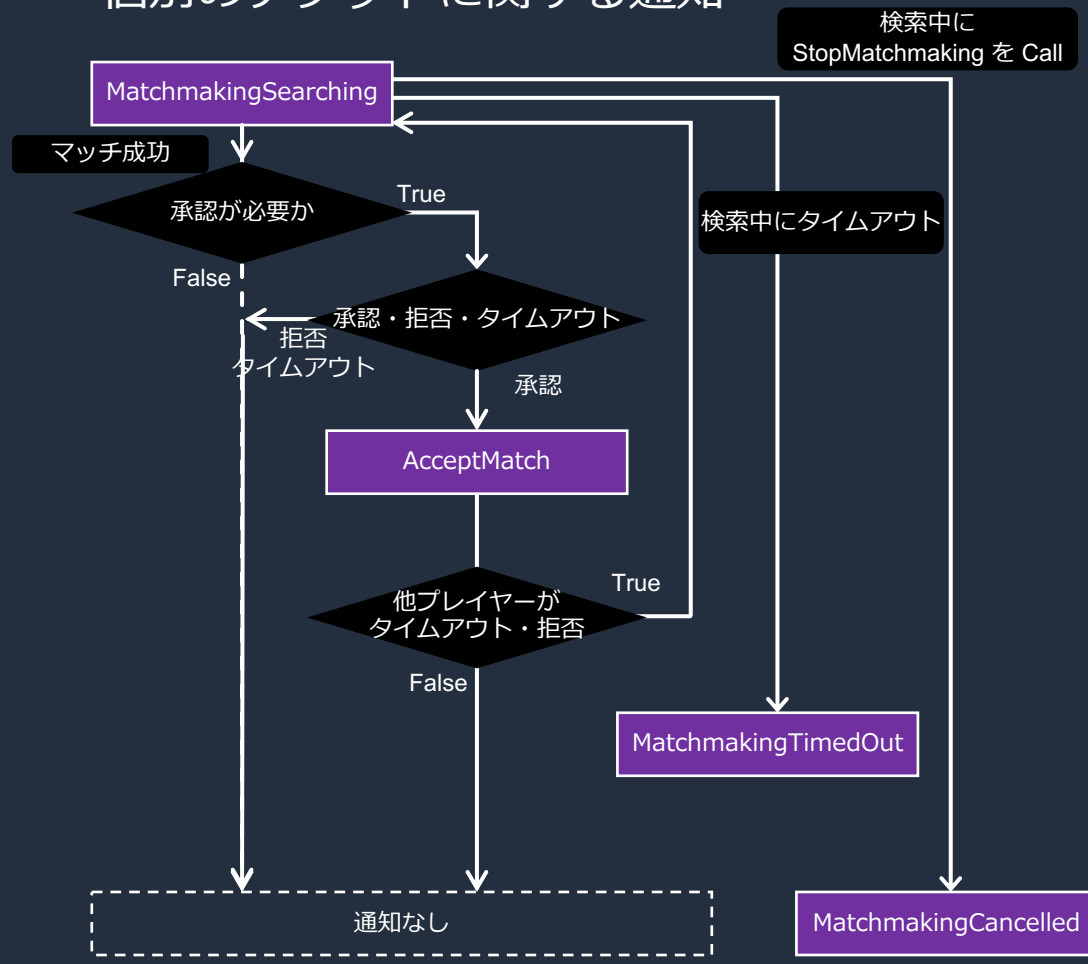
- PotentialMatchCreated　　：ルールセットの条件を満たすマッチの候補作成時に出力
- AcceptMatchCompleted　　：成否を問わずマッチ承諾プロセスが終了した際に終了
- MatchmakingSucceeded　　：マッチが成功した場合に出力
- MatchmakingFailed　　：エラー発生時に出力

通知へ出力されるステータスの順序関係全体図 (WITH_QUEUE かつ 承認必要)

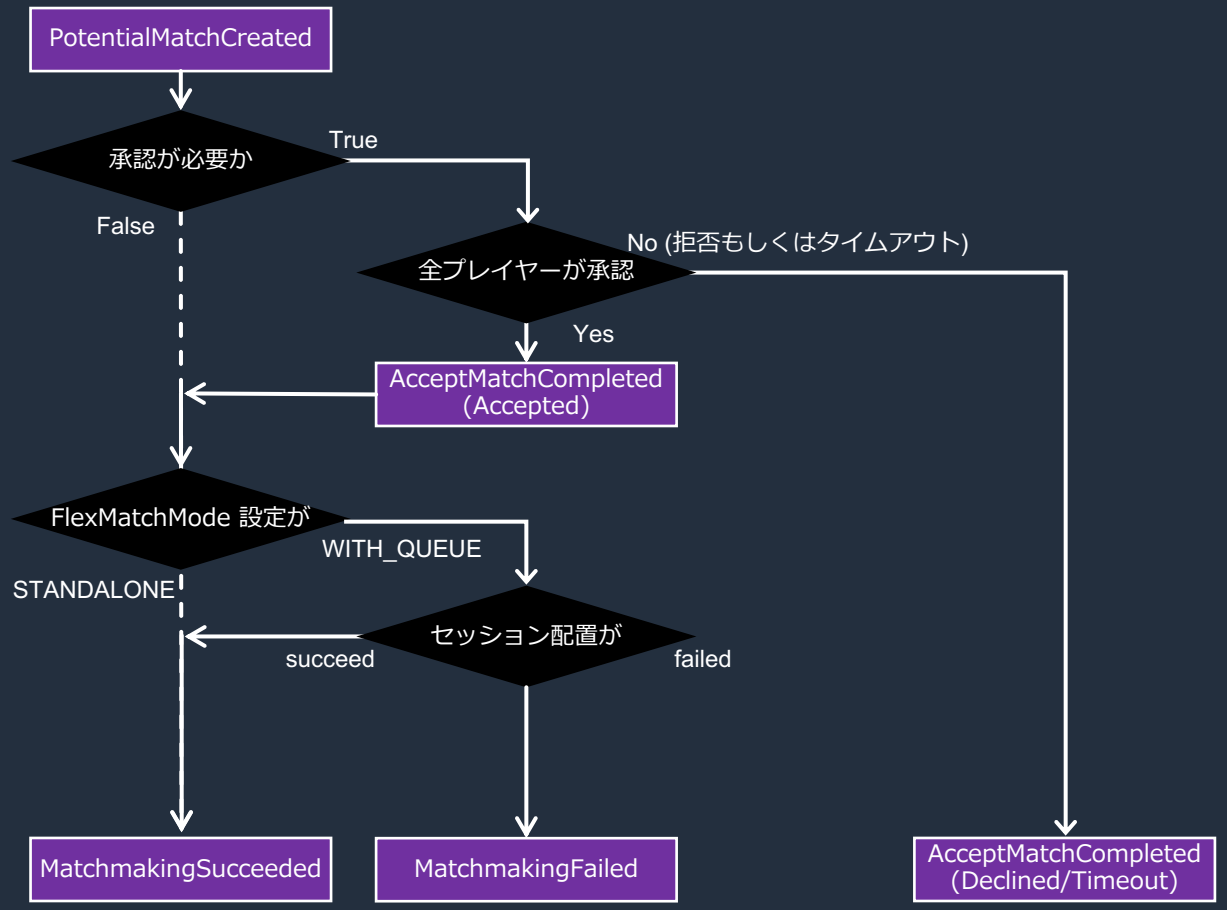
アクションもしくはイベント

通知タイプ

個別のチケットに関する通知



マッチに関する通知

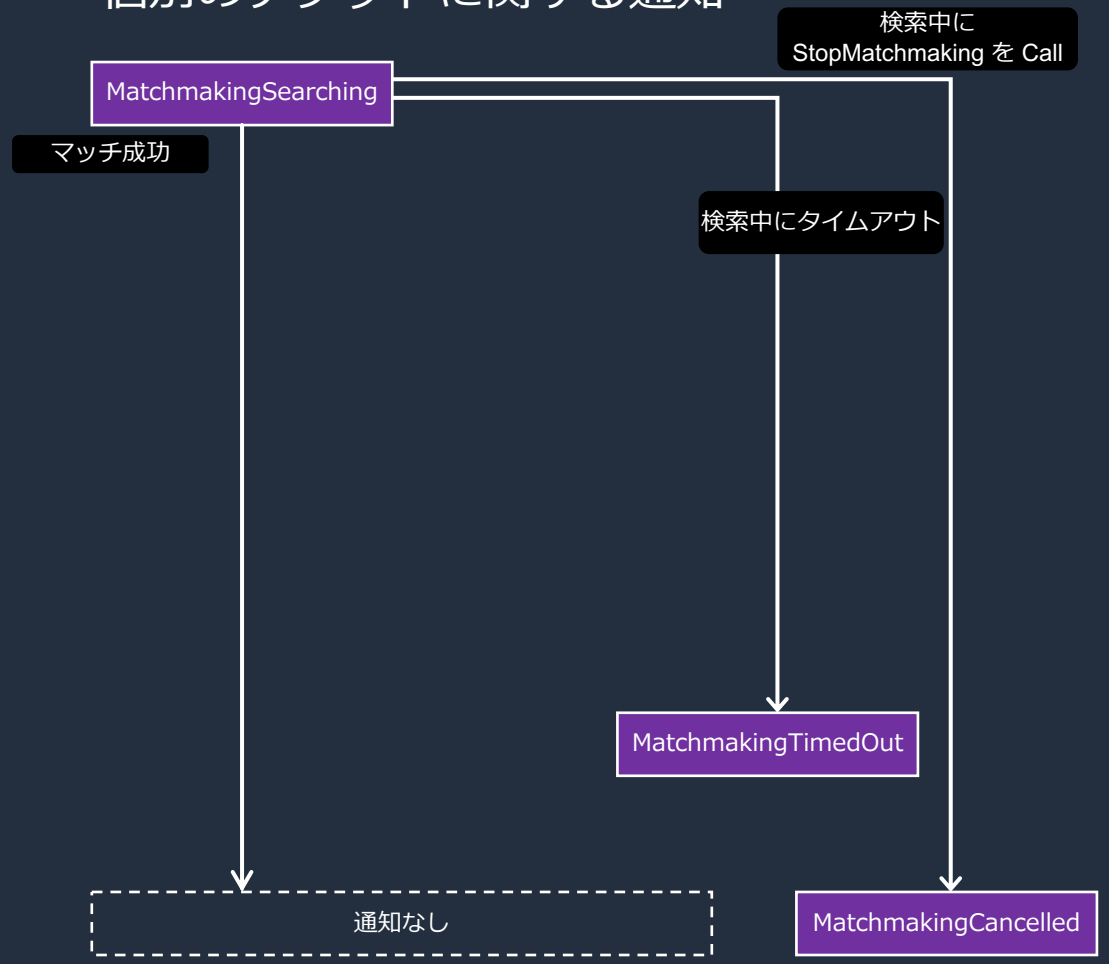


通知へ出力されるステータスの順序関係 (WITH_QUEUE かつ 承認不要)

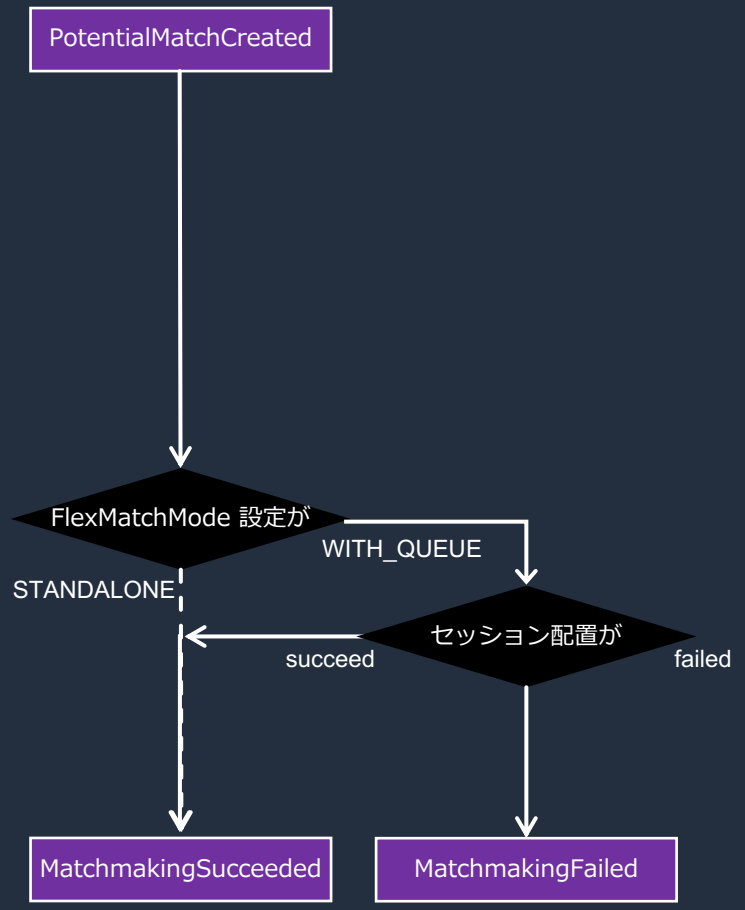
アクションもしくはイベント

通知タイプ

個別のチケットに関する通知



マッチに関する通知

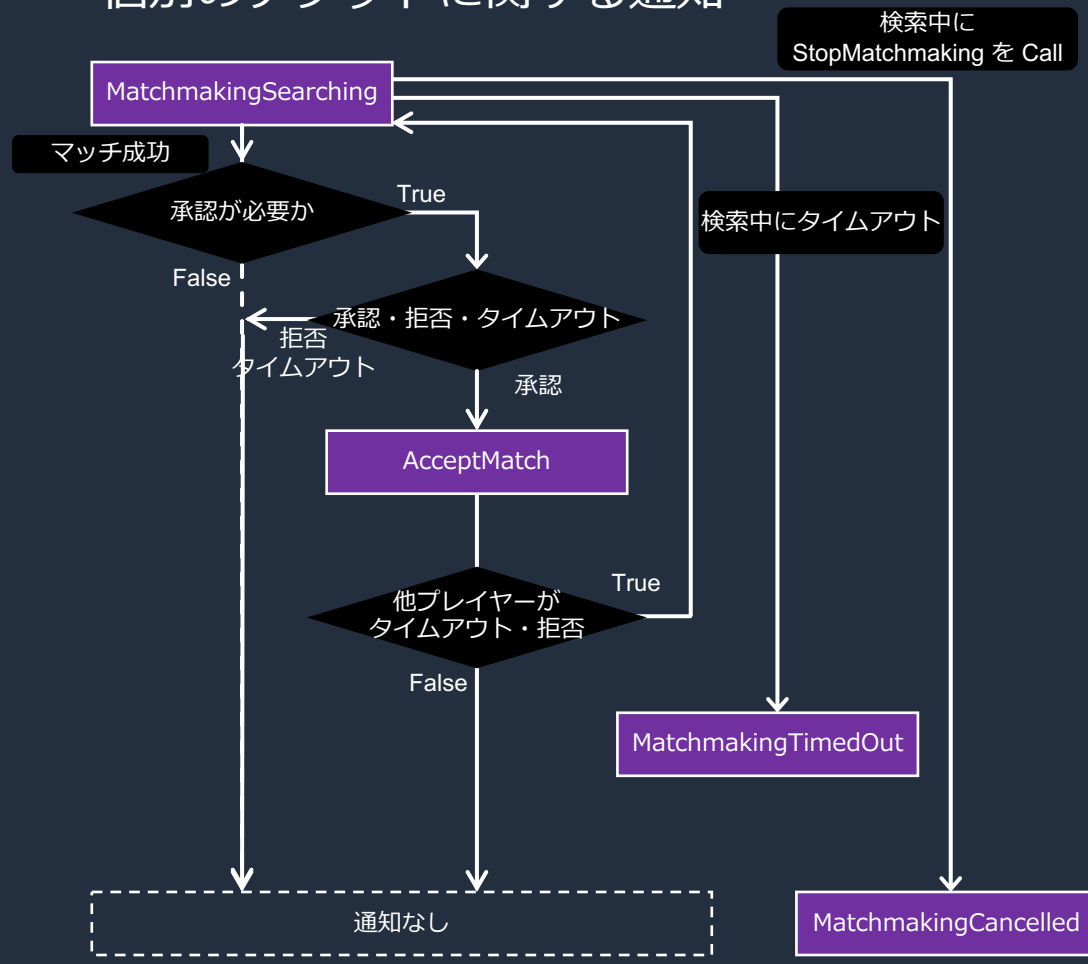


通知へ出力されるステータスの順序関係 (STANDALONE かつ 承認必要)

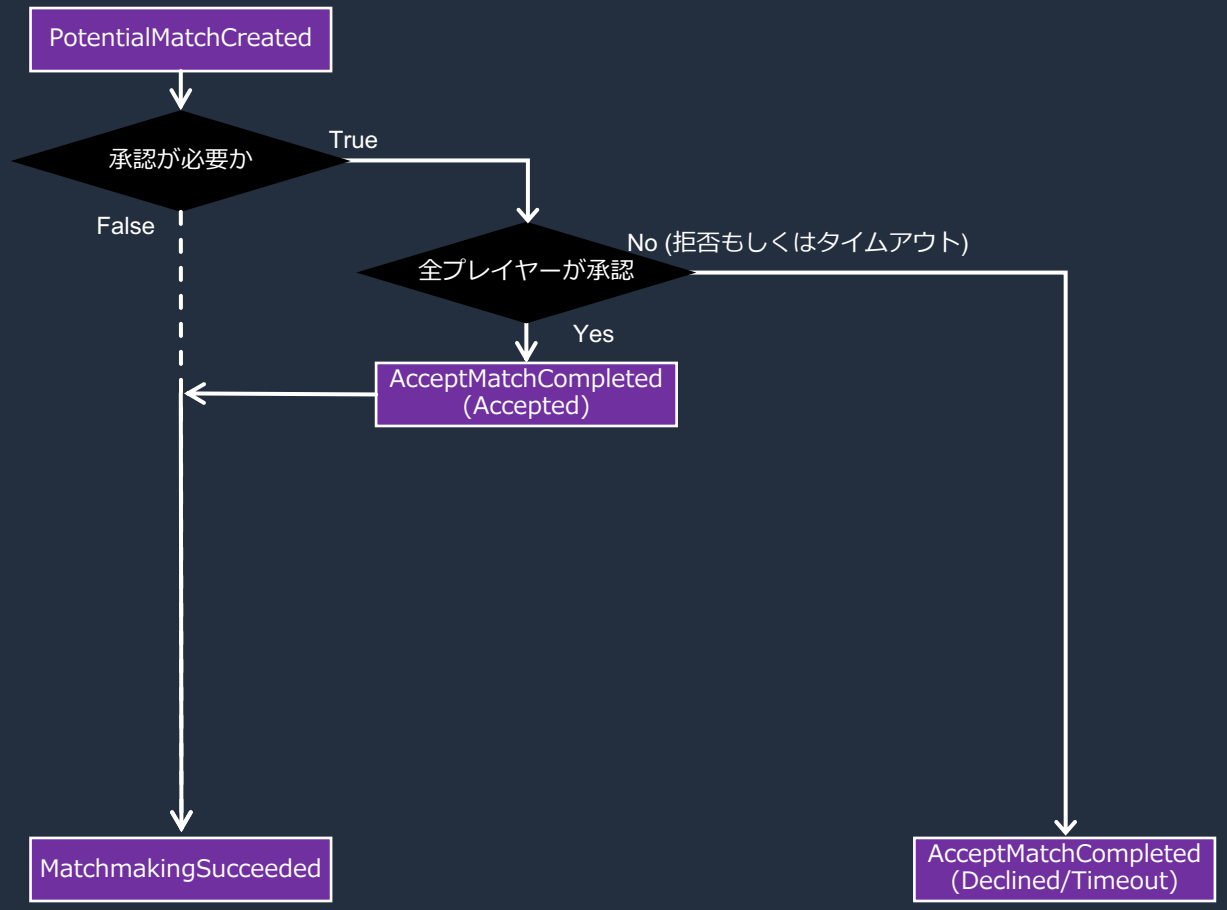
アクションもしくはイベント

通知タイプ

個別のチケットに関する通知



マッチに関する通知

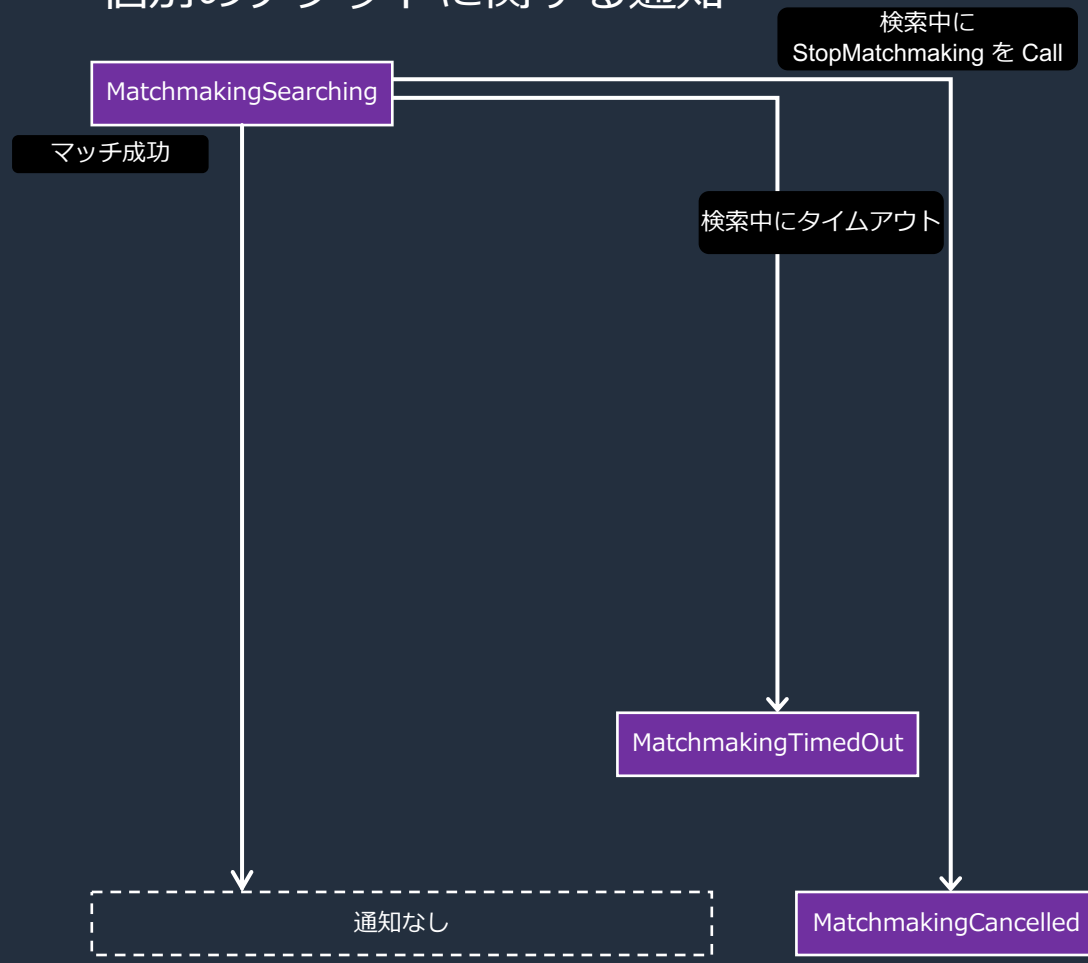


通知へ出力されるステータスの順序関係 (STANDALONE かつ 承認不要)

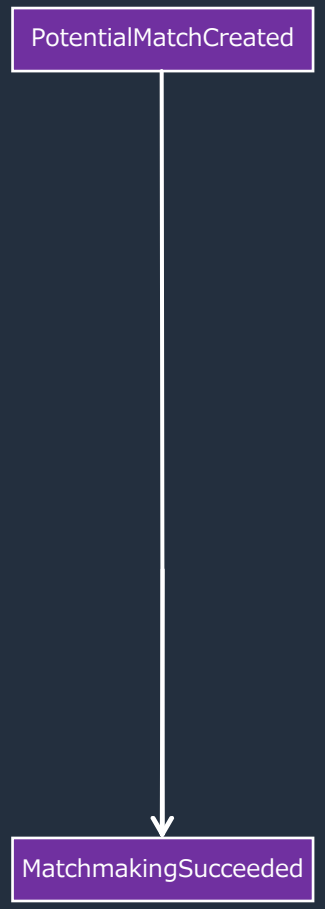
アクションもしくはイベント

通知タイプ

個別のチケットに関する通知



マッチに関する通知



料金

GameLift FlexMatch の料金

GameLift Managed と連携する場合、 FlexMatch の利用コストは発生しない
FlexMatch を単独で利用する場合、以下のコストが発生

	プレイヤーパッケージ	マッチメイキング時間
定義	StartMatchmaking API または StartMatchBackfill API で FlexMatch に送信されたプレイヤーの総数	FlexMatch のマッチごとにマッチ生成にあたってかかった時間を合計し、1時間単位に換算したもの
費用（東京リージョン）	$20.00 (100万プレイヤーパッケージあたり)	$1.00 (1マッチメイキング時間あたり)
計測方法	PlayersStarted メトリクス	MatchmakingSearchTime メトリクス
無料利用枠	50,000 プレイヤーパッケージ/月	マッチメイキング時間 5 時間/月

AWS Pricing Calculator で試算が可能

2023/07 時点
https://aws.amazon.com/jp/GameLift/pricing/

GameLift FlexMatch の料金

試算の例

- DAU（Daily Active User）が100,000 人を想定、月あたりの日数を３０日と仮定
- ５ 対 ５ のマッチメイキング評価に平均 50 ミリ秒消費
- １ゲーム平均10分のゲームを１ユーザーが平均3回プレイ

プレイヤーパッケージ料金

月あたりのプレイヤーパッケージ	DAU × ひと月の日数×ユーザーの平均プレイ回数
請求対象となるプレイヤーパッケージ	月あたりに発生したプレイヤーパッケージ – 無料利用枠
月当たりの利用料	月あたりのプレイヤーパッケージ数× プレイヤーパッケージあたりの料金

試算例

月あたりのプレイヤーパッケージ	100,000 × 30 × 3 = 9,000,000
請求対象となるプレイヤーパッケージ	9,000,000 – 50,00 = 8,950,000
月当たりの利用料	8,950,000 × 0.00002 = 179 (USD)

マッチメイキング時間料金

合計マッチ形成数	月当たりのプレイヤーパッケージ / マッチあたりのプレイヤー数
合計マッチメイキング時間	合計マッチ形成数×マッチあたりのマッチメイキング時間
請求対象マッチメイキング時間	合計マッチメイキング時間 – 無料利用枠
月当たりの利用料	請求対象マッチメイキング時間 ×時間あたりの料金

試算例

合計マッチ形成数	9,000,000 / 10 = 900,000
合計マッチメイキング時間	900,000 × (50 / 3,600,000) = 12.5
請求対象マッチメイキング時間	12.5 – 5.0 = 7.5
月当たりの利用料	7.5 × 1.0 = 7.5 (USD)

GameLift FlexMatch の料金

試算の例

- １ゲーム平均10分のゲームを１ユーザーが平均3回プレイする
- 1日の DAU (Daily Active User) が10,000 を想定
- 5 対 5 のマッチメイキング評価に平均 50 ミリ秒消費する。
- ゲームサーバーのホスティングに GameLift Managed Hosting を使用し、GameLift FlexMatch を WITH_QUEUE モードで使用する

FlexMatch で発生したコストは費用として請求されない

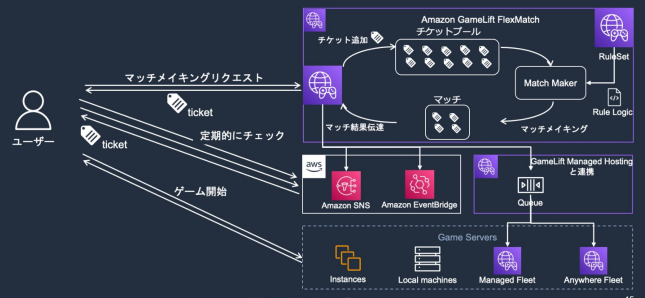
FlexMatch と連携した GameLift Managed Hosting 側のコストに注目

まとめ

まとめ

Amazon GameLift FlexMatch の全体像

FlexMatchはチケット発行、マッチ生成、他サービスへの通知を行う



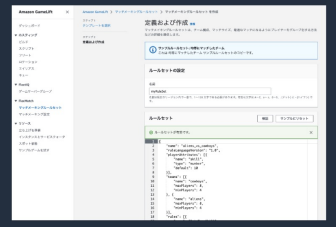
FlexMatch 使用開始の手順

4 stepでマッチメイキングを開始

Step1 マッチメイキングルールセットの作成

AWS コンソールでサンプルルールセットを選択

ルールセットの名前「myRuleSet」を入力し作成



マッチメイキング設定

マッチメイキングの動作設定や入出力などロジック以外の部分を設定

通知

- FlexMatch モード
- マッチメイキングのステータス通知

マッチメイキングの動作

- バックフィル
- マッチ承認
- タイムアウト

その他

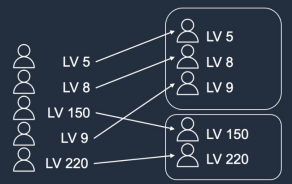
- マッチメイキング設定名
- カスタムデータ
- イベントデータ
- タグ

ルールセット概要

マッチメイキングの時のユーザー条件や構造を JSON で記述

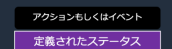
特徴

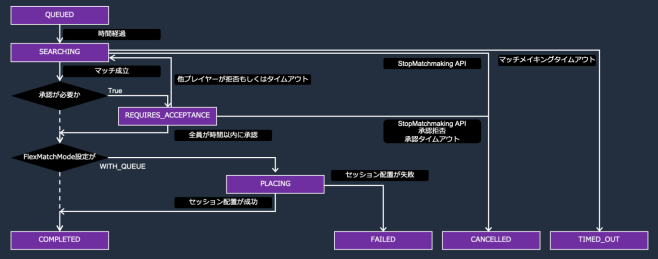
- 多様なチーム構造
- ユーザー属性に応じたマッチメイキング
- 時間経過による条件の緩和
- 少人数向け（スモールマッチ）と大人数向け（ラージマッチ）の 2 種類のモード



ユーザー属性に応じたマッチメイキングの例

チケットのステータス遷移全体図 (WITH_QUEUE かつ 承認必要)





GameLift FlexMatch の料金

GameLift Managed と連携する場合、 FlexMatch の利用コストは発生しない
FlexMatch を単独で利用する場合、以下のコストが発生

	プレイヤーパッケージ	マッチメイキング時間
定義	StartMatchmaking API または StartMatchBackfill API で FlexMatch に送信されたプレイヤーの総数	FlexMatch のマッチごとにマッチ生成にあたってかかった時間を合計し、1 時間単位に換算したもの
費用（東京リージョン）	$20.00 (100万プレイヤーパッケージあたり)	$1.00 (1マッチメイキング時間あたり)
計測方法	PlayersStarted メトリクス	MatchmakingSearchTime メトリクス
無料利用枠	50,000 プレイヤーパッケージ/月	マッチメイキング時間 5 時間/月

AWS Pricing Calculator で試算が可能

2023/07 時点
https://aws.amazon.com/jp/GameLift/pricing/

本資料に関するお問い合わせ・ご感想

技術的な内容に関しましては、有料のAWSサポート窓口へ
お問い合わせください

https://aws.amazon.com/jp/premiumsupport/

料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
お問い合わせください（マネジメントコンソールへのログインが必要です）

https://console.aws.amazon.com/support/home#/case/create?issueType=customer-service

具体的な案件に対する構成相談は、後述する個別相談会をご活用ください

ご感想はTwitterへ！ハッシュタグは以下をご利用ください
#awsblackbelt

その他コンテンツのご紹介

ウェビナーなど、AWSのイベントスケジュールをご参照いただけます

https://aws.amazon.com/jp/events/

ハンズオンコンテンツ

https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-hands-on/

AWS 個別相談会

AWSのソリューションアーキテクトと直接会話いただけます

https://pages.awscloud.com/JAPAN-event-SP-Weekly-Sales-Consulting-Seminar-2021-reg-event.html

Thank you!