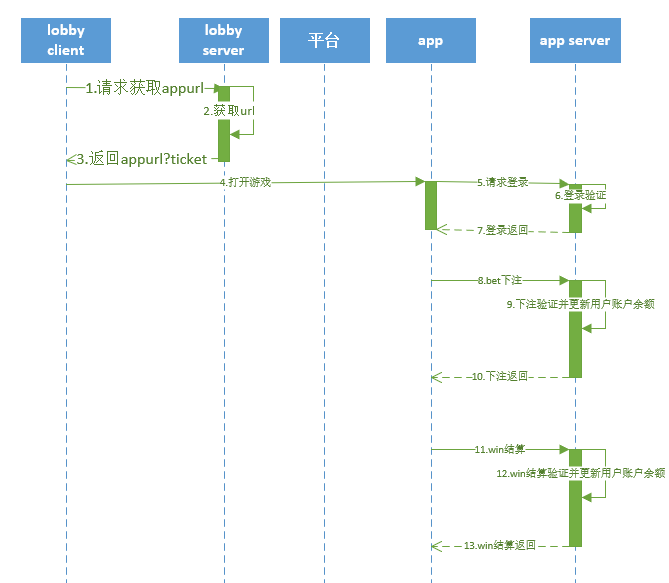
**平台UML流程文档**

1. 自有运营商和自有提供商



1. 大厅客户端向大厅服务端发起获取AppUrl的http请求。

var ipo = new AppUrlIpo()

{

UserId = userId,

AppId = "best\_shooter",

OperatorId = "own\_lobby\_mex",

CountryId = "MEX",

CurrencyId = "MXN",

LangId = "",

IsDemo = false

}

1. 大厅服务端通过XxyyProviderClient 获取appurl,同时在Redis中保存登录ticket和token，ticket用于游戏App登录验证使用,有效期30s,用完即清除，token用于游戏后端后续bet下注,win结算使用,有效期30min。
2. 大厅服务端向大厅客户端返回AppUrl?ticket并携带ticket。
3. 大厅客户端将用户重定向至返回的appurl打开游戏。
4. 用户请求登录游戏，携带ticket。
5. 游戏服务端验证当前登录是否合法，从Redis中获取当前ticket信息是否一致并保存token信息，用于后续bet下注和win结算使用，同时生成游戏访问的jwt-token。
6. 游戏服务端返回登录结果。
7. 用户在游戏中向游戏服务端请求bet下注。

BetWinIpo ipo = new BetWinIpo()

{

RequestUUID = Guid.NewGuid().ToString(),

TransactionUUID = transactionUUID,

UserId = \_userId,

AppId = \_bestShooterOptionsSection.AppId,

CurrencyId = \_appUserDCache.CurrencyId,

Token = \_appUserDCache.PlatformToken,

RoundClosed = true,

RoundId = string.Empty,

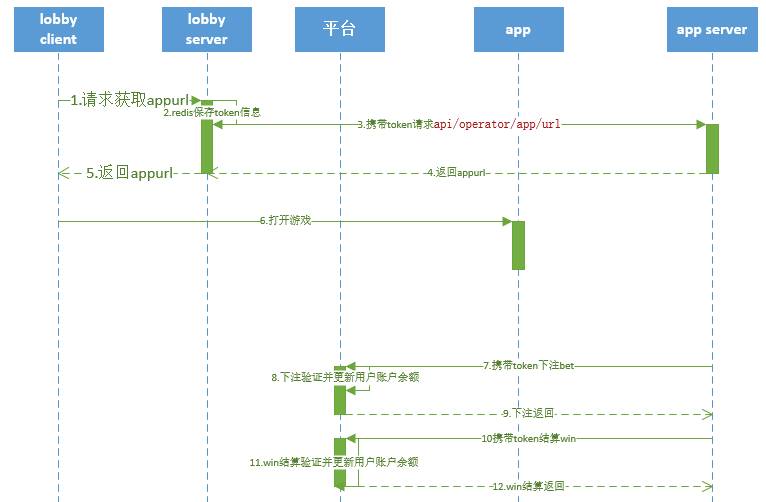
RewardUUID = rewardUUID,

IsFree = false,

Amount = amount

}

1. 游戏服务端验证下注是否合法并更新用户账户余额。
2. 游戏服务端返回下注结果。
3. 自有运营商和第三方提供商



1. 大厅客户端向大厅服务端发起获取AppUrl的http请求。

var ipo = new AppUrlIpo()

{

UserId = userId,

AppId = "best\_shooter",

OperatorId = "own\_lobby\_mex",

CountryId = "MEX",

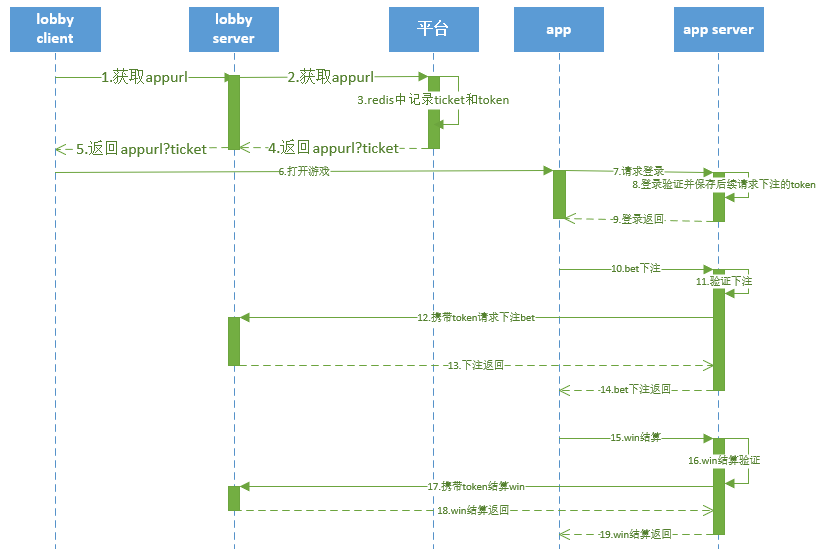
CurrencyId = "MXN",

LangId = "",

IsDemo = false

}

1. 大厅服务端收到请求大厅客户端请求后在Redis保存第三方提供商后续请求下注使用的token。
2. 大厅服务端通过XxyyProviderClient引用方式在OperatorService类中AppUrl方法中 创建ProviderProxy，通过ProviderProxy中AppUrl方法向第三方提供商接口api/operator/app/url发送http请求获取appurl,同时在Redis中保存token，token用于游戏后端后续向平台bet下注,win结算使用,有效期30min。
3. 第三方提供商向大厅服务端返回appurl。
4. 大厅服务端向大厅客户端返回appurl。
5. 大厅客户端将用户重定向至返回的appurl打开游戏。
6. 第三方游戏服务端携带token向平台发送http 请求bet下注。
7. 平台验证bet下注请求是否合法并更新用户账户余额。
8. 平台返回下注结果。
9. **第三方运营商和自有提供商**



1. 大厅客户端向大厅服务端发起http请求获取游戏appurl。
2. 大厅服务端向平台/api/operator/app/url发送http请求。
3. 平台到请求,平台验证第三方运营商的http请求签名,平台根据自有提供商直接获取appurl,同时在Redis中保存登录ticket和token，ticket用于游戏App登录验证使用,有效期30s(秒),用完即清除，token用于游戏后端后续bet下注,win结算使用,有效期30min(分钟)。

var ipo = new AppUrlIpo()

{

UserId = userId,

AppId = "best\_shooter",

OperatorId = "own\_lobby\_mex",

CountryId = "MEX",

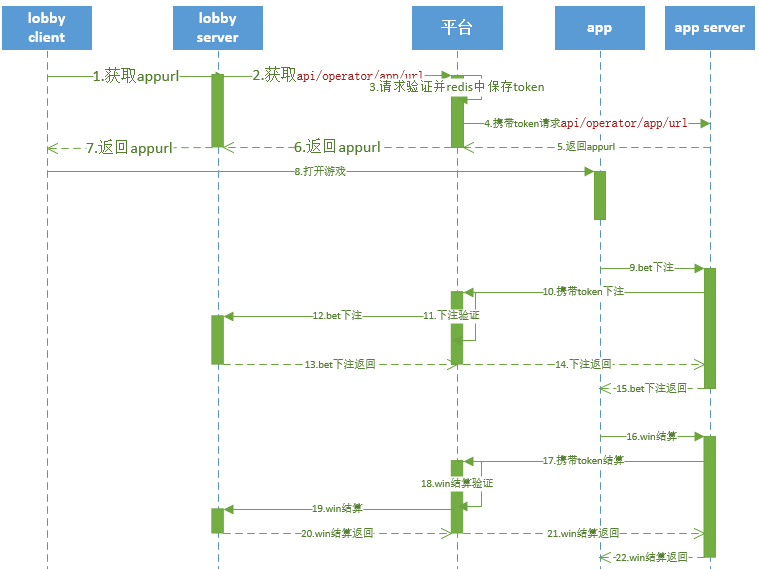
CurrencyId = "MXN",

LangId = "",

IsDemo = false

}

1. 平台向第三方大厅服务端返回appurl。
2. 大厅服务端向大厅客户端返回appurl。
3. 大厅客户端将用户重定向至获取的appurl。
4. 用户向游戏服务端发送http登录请求携带ticket。
5. 游戏服务端验证登录请求，从Redis中获取ticket进行验证并颁发用于登录游戏的jwt-token。
6. 游戏服务端返回登录结果。
7. 用户向游戏服务端请求下注bet。
8. 游戏服务端验证下注请求。
9. 游戏服务端通过XxyyOperatorClient引用方式在ProviderService类中Bet方法创建OperatorProxy，并向大厅服务端发送http下注bet请求。
10. 大厅服务端向游戏服务端返回下注bet结果。
11. 游戏服务端向用户返回下注bet结果。
12. **第三方运营商和第三方提供商**



1. 大厅客户端向大厅服务端发送http获取appurl。
2. 大厅服务端收到http请求向平台/api/operator/app/url发送http请求。

var ipo = new AppUrlIpo()

{

UserId = userId,

AppId = "best\_shooter",

OperatorId = "own\_lobby\_mex",

CountryId = "MEX",

CurrencyId = "MXN",

LangId = "",

IsDemo = false

}

1. 平台收到大厅服务端appurl请求,验证请求并在Redis中保存token，用于后续bet下注和win结算等http请求验证使用。
2. 平台通过OperatorService类中AppUrl方法创建ProviderProxy，在ProviderProxy类中AppUrl方法中携带token向游戏服务端/api/operator/app/url发送http请求获取AppUrl。
3. 游戏服务端验证appurl请求并向平台返回AppUrl。
4. 平台向大厅服务端返回AppUrl。
5. 大厅服务端向大厅客户都返回AppUrl。
6. 大厅客户都将用户重定向至获取的AppUrl。
7. 用户向游戏服务端发送bet下注请求。
8. 游戏服务端携带token向平台/api/provider/transation/bet发送http请求bet下注。
9. 平台收到http下注请求后验证当前下注bet请求。
10. 平台向大厅服务端发送http请求下注。
11. 大厅服务端收到bet下注请求并验证当前请求以及更新用户账户余额，返回更新后的账户余额并向平台返回。
12. 平台收到大厅服务端返回并向游戏服务端返回bet下注结果。
13. 游戏服务端收到平台bet下注返回向用户返回bet下注结果。