Sportsbook in iframe

Brief Description

This service provides external consumers with access to a sportsbook.

The provider is a betting platform that provides the online sportsbook to the merchant.

The merchant is a legal entity that has entered into or is planning to enter into an agreement with the provider for the provision of betting services and/or the provision of related services.

The sportsbook includes the following functionality:

- Information on upcoming events
- Information on live broadcasts
- Information on betting history
- Information on event results
- Information on statistics
- Placing bets

To integrate the service, you need to configure the API and iframe

The provider gives the merchant operator and operatorKey during the integration development process

- operator is a merchant's identifier in provider's system
- operatorKey is a secret key, used in hashing by both merchant and provider

Serialization Requirements:

1. encoding: UTF-8
2. property names are compared case-insensitively
3. naming policy is camel case. Example: textForExample
4. Enumerations are serialized as strings
5. fields with missing values (null or 0) are not ignored and serialized with default values
6. number handling:

- reading from strings is allowed. Example: "field": 42 and "field": "42" will be deserialized as the number 42
- currency names must be represented using their alphabetic code, according to the ISO 4217 standard. Example: USD, CLP, RUB
- currencies that have minor units must be transmitted in minor units. Example: 100.12 USD = 10012, 100.00 RUB = 10000
- currencies that do not have minor units are transmitted in standard values. Example: 100 CLP = 100, 100 JPY = 100
- the number of minor units is determined by the ISO 4217 standard

Response codes:

A successful response contains the "Ok" code

Errors:

- Undefined
- InternalError
- InvalidRequest
- InvalidHash
- UserNotFound
- InsufficientFunds
- MaxBetAmountExceeded
- BetNotFound
- SessionTokenNotFound
- InvalidCurrency

If an error occurs, return a JSON object ErrorResponse

ErrorResponse

№	Name	Data Type	Nullability
1	ErrorCode	string	not null
2	ErrorMessage	string	null

Generation hash:

The hash is generated using the hmac-sha256 algorithm. The message is a concatenation of the method name and fields without spaces. The key used is the operatorKey.

Authenticate: "authenticate{Token}", getbalance: "getbalance{userId}{currency}", changebalance: "changebalance{betId}{userId}{currency}{amount}{transactionType}"

Service Methods:

Authorization and Authentication:

- For authorized users, a token must be generated and inserted into the URL. Example URL: https://provider.com/live?session_token={token}&operator={operator}&hash={hash}
- Unauthorized users do not require a token. However, they will not have access to the betting functionality.

Use case:

Precondition: The user may or may not be authenticated.

Trigger: The user navigates to a page with an iframe.

Steps:
1. The merchant checks the user's authentication status.

  a. If the user is authenticated:
    i. The merchant generates a session token
    ii. The merchant inserts the session token and operator name into the URL. Example URL: https://provider.com/live?session_token={token}&operator={operator}&hash={hash}
    iii. The merchant calculates hash from the session token and operator as: hash(message, secretKey), where message is concatenation of session_token and operator ("{session_token}{operator}"), secretKey is operatorKey, given to the merchant by provider
    iv. The provider sends a request to the merchant (AuthenticateRequest)
    v. The merchant sends user information to the provider (AuthenticateResponse)
    vi. The provider authenticates the user
    vii. The provider grants the user access to betting

  b. If the user is not authenticated:
    i. The merchant displays the provider's website without betting functionality. Example URL: https://provider.com/live

Authenticate

Method	POST /api/authenticate
Input data	AuthenticateRequest
Output data	AuthenticateResponse

AuthenticateRequest

№	Name	Data Type	Nullability	Description
1	Token	string	not null	Session token generated by merchant
2	Hash	string	not null	-

AuthenticateResponse

№	Name	Data Type	Nullability	Description
1	UserId	string (64)	not null	Unique user ID in merchant's system
2	Username	string (100)	not null	User display name. Duplicates are allowed
3	Currency	string(3)	not null	User currency (ISO 4217 alphabetic code)

Player balance adjustment:

Use case:

Precondition: The user is authenticated.

Trigger: The user places a bet.

Steps:
1. The provider sends a user balance request (GetBalanceRequest).
2. The merchant sends the user’s balance (GetBalanceResponse).

    a. If the user has sufficient funds:
        i. The provider sends a balance change request along with bet details (ChangeBalanceRequest).
        ii. The merchant updates the user’s balance and sends transaction details (ChangeBalanceResponse).
        iii. The provider processes the bet.

    b. If the user has insufficient funds:
        i. The provider rejects the bet.

Get Balance

Method	POST /api/getbalance
Input data	GetBalanceRequest
Output data	GetBalanceResponse

GetBalanceRequest

№	Name	Data Type	Nullability	Description
1	UserId	string(64)	not null	Unique user ID in merchant's system
2	Currency	string(3)	not null	User currency (ISO 4217 alphabetic code)
3	Hash	string	not null	-

GetBalanceResponse

№	Name	Data Type	Nullability	Description
1	Balance	int (long)	not null	User balance
2	Currency	string (3)	not null	User currency (ISO 4217 alphabetic code). User's currency must not differ in subsequent requests

Change Balance

Method	POST /api/changebalance
Input data	ChangeBalanceRequest
Output data	ChangeBalanceResponse

ChangeBalanceRequest

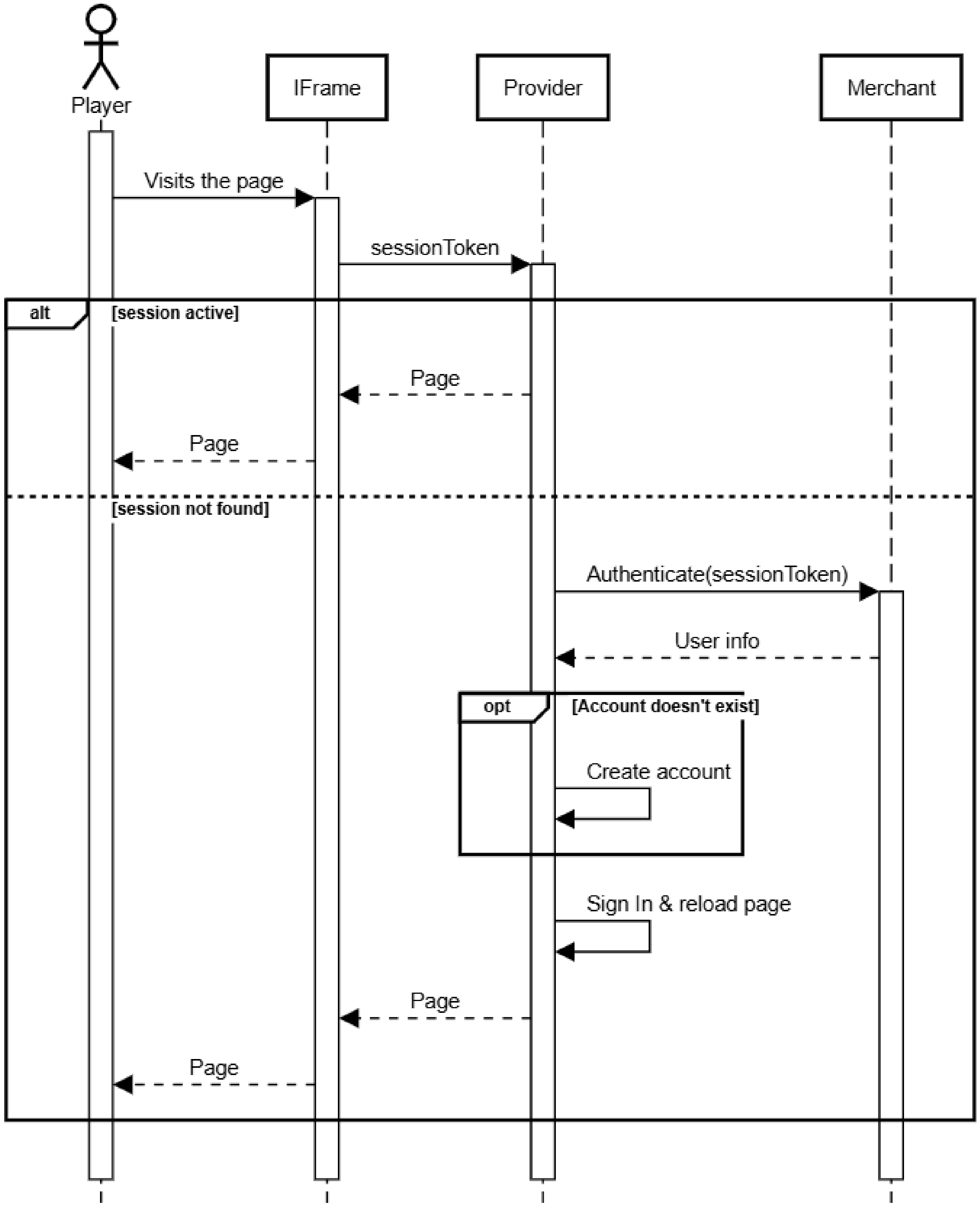
№	Name	Data Type	Nullability	Description
1	BetId	int (64)	not null	Bet ID in provider's system
2	UserId	string (64)	not null	Unique user ID in merchant's system
3	Currency	string (3)	not null	User currency (ISO 4217 alphabetic code)
4	Amount	number	not null	Transaction amount. The value is always greater than 0
5	TransactionType	enum(string)	not null	Transaction type: • BetPlacement - debit of funds to place a bet • BetRefund - depositing funds due to bet refund • BetWin - depositing funds due to bet win • BetRecalculation - debit of funds due to recalculation of the bet
6	Hash	string	not null	-

ChangeBalanceResponse

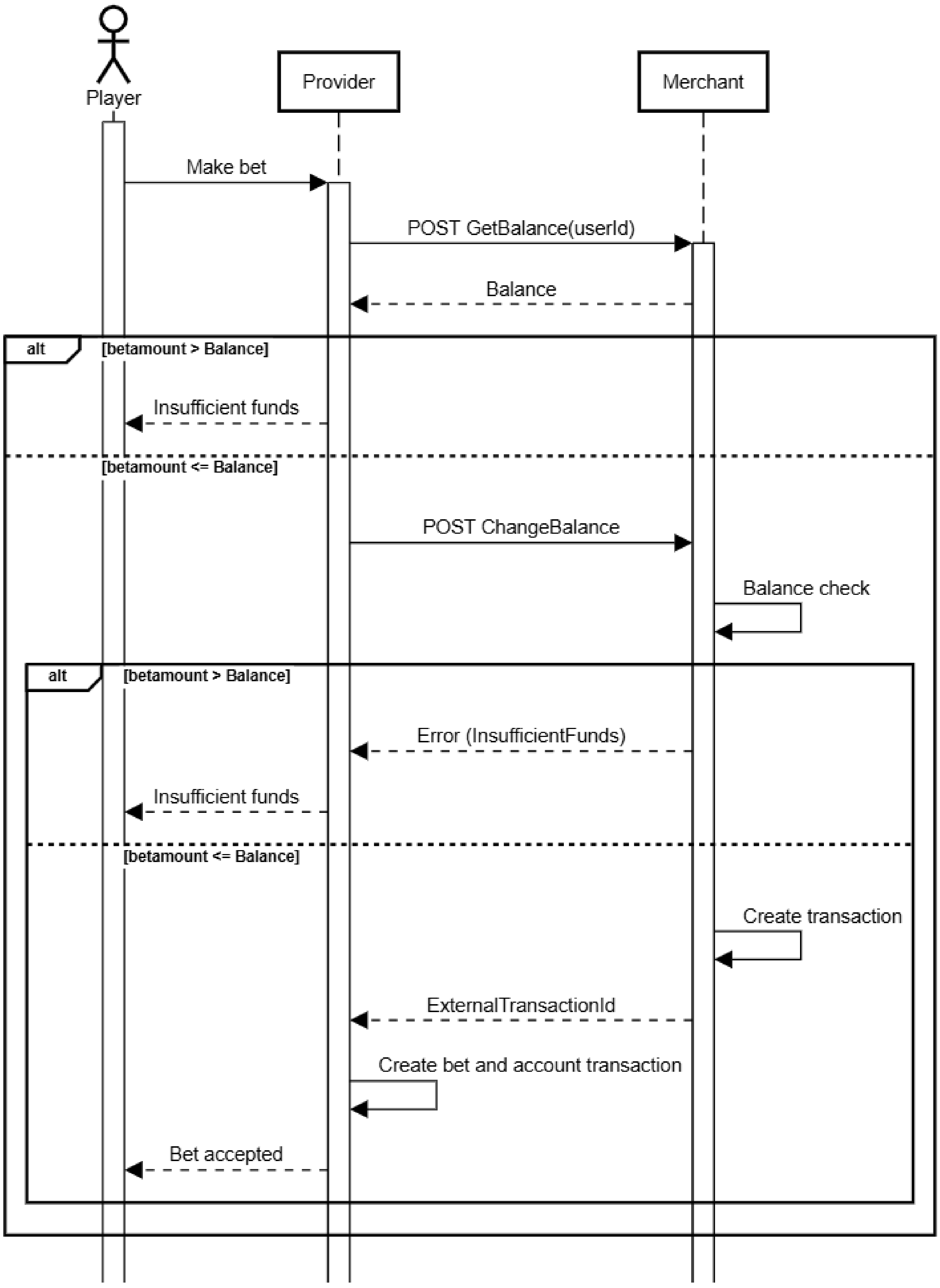
№	Name	Data Type	Nullability	Description
1	Balance	int (long)	not null	User balance
2	Currency	string (3)	not null	User currency (ISO 4217 alphabetic code). User's currency must not differ in subsequent requests
3	TransactionId	string (64)	not null	Unique transaction ID in merchant's system

Sequence diagrams

Authentication



Change balance


Player
Provider
Merchant
Make bet
POST GetBalance(userId)
Balance
alt
[betamount > Balance]
Insufficient funds
[betamount <= Balance]
POST ChangeBalance
Balance check
alt
[betamount > Balance]
Error (InsufficientFunds)
Insufficient funds
[betamount <= Balance]
Create transaction
ExternalTransactionId
Create bet and account transaction
Bet accepted