

# Directive XX.2023: Betting Monitoring and Reporting System

This document contains the requirements with which bookmakers must comply when reporting betting data to the National Betting Authority of Cyprus.

## API Data Requirements - User's Guide

## Version History

Version	Date	Description
1.0	7/8/2023	-

# Table of Contents

## Contents

Table of Contents.....	3
<b>Directive XX.2023: Betting Monitoring and Reporting System.....</b>	<b>4</b>
<b>Annex A: Technical Requirements.....</b>	<b>5</b>
1.0 Introduction.....	5
<b>2.0 The BMRS Data Model.....</b>	<b>6</b>
2.1 Bookmakers, Authorized Representatives and Shops (Licensees).....	7
2.2 Players and Accounts.....	7
2.3 The Betting Slip.....	8
<b>3.0 The BMRS API Processes.....</b>	<b>10</b>
3.1 Player Accounts.....	12
3.1.1 Method SaveAccounts.....	12
3.2 Account Transactions.....	18
3.2.1 Method CreateAccountTransactions.....	18
3.3 Account Restrictions.....	24
3.3.1 Method SaveAccountRestrictions.....	25
3.4 Betting Activity.....	30
3.4.1 Method CreateBetSlips.....	31
3.4.3 Method UpdateBetSlips.....	42
3.4.5 Method SaveBetLines.....	51
3.5 System Calls.....	55
3.5.1 Method Heartbeat.....	55
3.5.2 Method GetErrorLogs.....	56
4.0 BMRS Communication Protocols.....	60
4.1 Primary Communication Protocol – The BMRS API.....	60
4.2 Secondary Communication Protocol – XML Files.....	62

## **Directive XX.2023: Betting Monitoring and Reporting System**

This Directive is drafted and issued by the National Betting Authority (henceforth “NBA”) according to the provisions of paragraph (b) of article 15 of the Betting Act 2009 (L.37(I)/2019), as amended or replaced, and sets out all the technical requirements and procedures that Class A and B Licensees must adopt in order to connect and transmit information to the NBA via the NBA’s Betting Monitoring and Reporting System (henceforth “BMRS”). It prescribes the types of data that must be transmitted as well the instances or timeframes that trigger the transmission.

2. All Class A and B Licensees must –

- a) adopt and implement all the technical requirements and procedures of Annex A,
- b) connect to the BRMS as prescribed in Annex A,
- c) transmit data at times and instances as prescribed in Annex A.

3. This Directive is binding and its implementation is mandatory for all Class A and B Licensees.

## Annex A: Technical Requirements

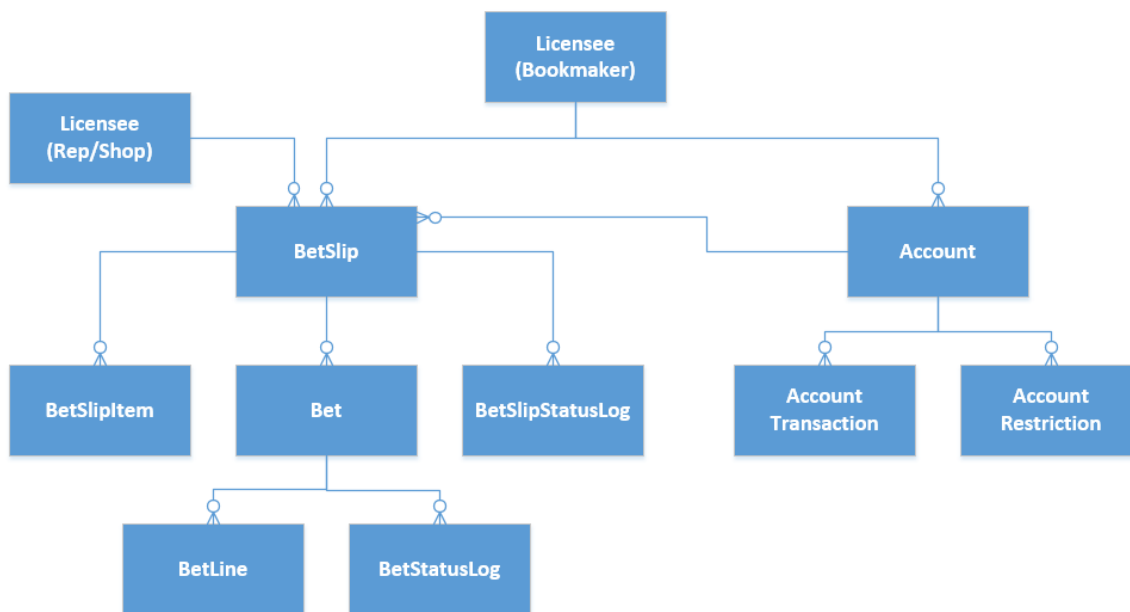
### 1.0 Introduction

The BMRS is a monitoring platform developed by the NBA as part of enhancing its data-centric regulating frame by closely following the activity of all participants of the betting market in near real-time. Bookmakers that facilitate betting activities either online (Class B Licensees) or through authorized representatives and retail shops (Class A Licensees) are expected to engage in continuous communication with the platform through a suitable designed API and transmit all information the NBA deems necessary towards fulfilling its regulatory mandate.

This document describes in detail the different processes the system makes available to the end-users (licensees), the purpose and data fields each of these processes requires, the expected form to which data transmissions need to conform (XML schemas), responses and potential errors arising from data validation as well as a general description of the communication protocols employed.

## 2.0 The BMRS Data Model

The BMRS collects and processes data in a way that allows both for an insightful overview of market trends and behaviours and, where required, a detailed breakdown of betting activities at the individual level. Its design is based on the modelling of certain key entities and the processes by which they interact with one another to collectively shape the operations of the betting business. An outline of the model is shown in Figure 2.1.



**Figure 2.1:** Overview of the BMRS data model.

To each bet there are two counterparties, namely the bookmaker (licensee) that offers certain odds for a particular outcome of a future underlying event, and the player who accepts those odds and decides to place a certain amount of stake towards them. All information with regards to a particular bet is contained on a betting slip, which becomes a valid form of agreement once submitted by the player and accepted by the bookmaker. Thereafter, the bet slip follows a certain lifecycle, which is terminated either once all events on which the player has bet on are concluded without a payout, or upon the payment of any realised winnings (or refund).

## 2.1 Bookmakers, Authorized Representatives and Shops (Licensees)

**Bookmakers** facilitate betting activities either online (Class B) or through **authorized representatives** and retail **shops** (Class A). In the former case, they provide platforms through which players can register an **account** and subsequently submit **betslips**. In the case of a land-based operation, the players do not register their details through an account but submit their bets through the shop cashiers.

Because all three of these entities are regulated under law, details regarding their structure and operational setup are tracked by a number of systems outside the BMRS. During data exchanges through the methods listed in Section 3.0, the BMRS uniquely identifies each sender but the NBA does not rely on this particular communication channel for managing the detailed records of these entities.

## 2.2 Players and Accounts

In the case of online betting, a person may only initiate a wager after registering an **account** with a Class B licensed bookmaker. The account, which is maintained on the bookmaker's platform, holds certain identification details of its owner and keeps a record of the funds available to them for bet placement. The bookmaker's platform facilitates a number of **transactions** that affect the account balance, such as the deposit of additional funds by the account owner, the withdrawal of funds and the tracking of any bonus that the bookmaker might have offered to the player under certain rules and conditions. Further to the above, a player is also provided with the option to impose certain limits on their betting activities by means of **restrictions**. In some cases, **restrictions** may also be applied on a player's account by the bookmaker for suspicion of any rule violation.

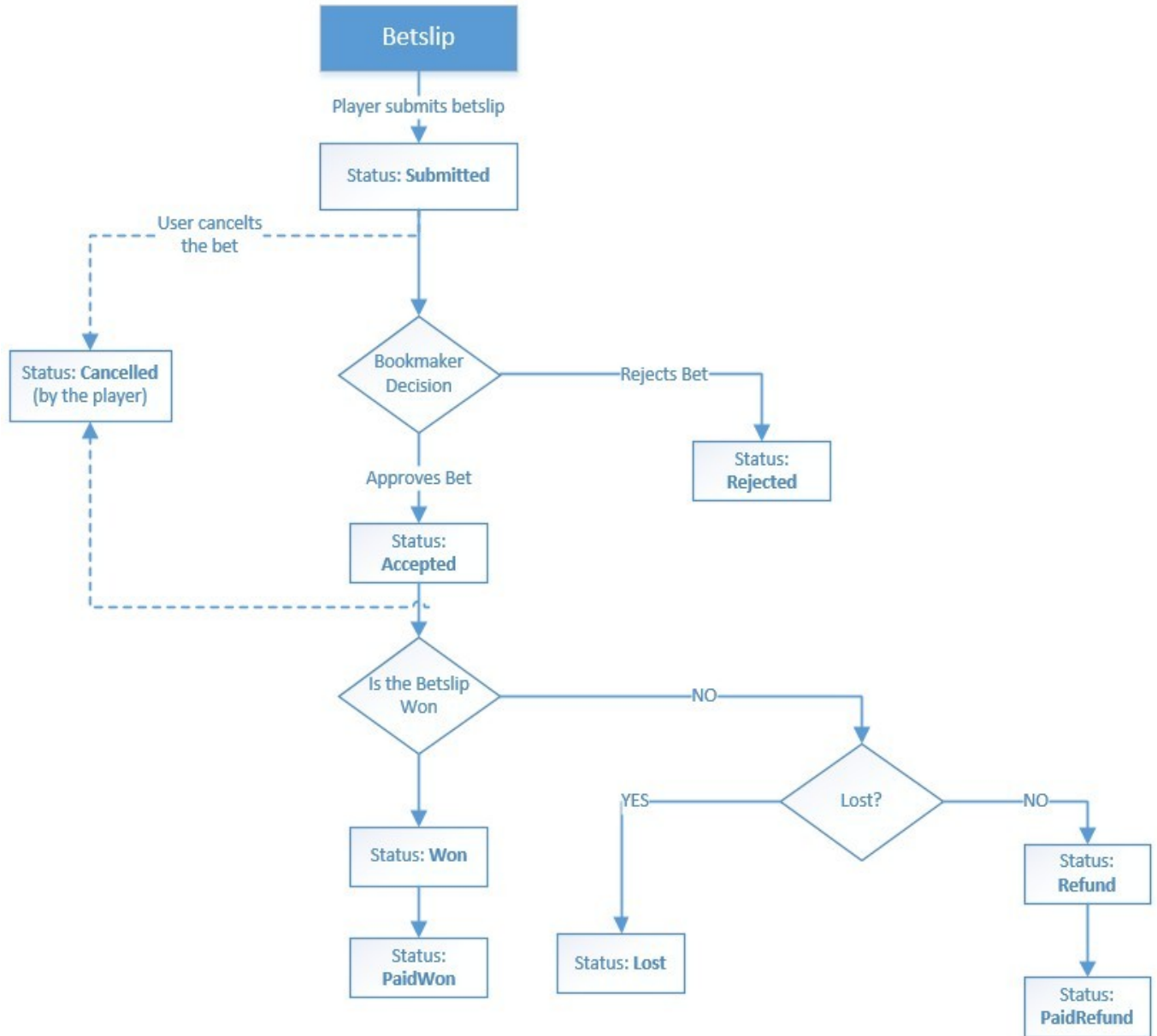
In land-based betting operations, the player does not currently need an account to participate in wagering, and most of what was described in the previous paragraph is not relevant. Within the

BMRS model, a player's account in the context of retail betting represents the cashier of the shop through which a bet slip was submitted.

## 2.3 The Betting Slip

At the root of the BMRS data model employed to represent betting traffic is the betting slip (**betslip**). A betting slip may be issued at a shop in the case of a Class A type license or online in the case of Class B. It is comprised of one or more **betslip items**, each of which contains information on a certain underlying event, a market associated with that event and the outcome of that market (selection) that the player wishes to place a bet on against the odds offered by the bookmaker. A **bet**, on which the player places an amount of stake, may involve one or more of these betslip items (events, markets, and player selections) which together form a set of possible combinations or **bet lines**. Because each of these combinations is made up of different constituent items, the odds differ from one line to the next and thus a bet could be assigned a range of offered odds.

Once created, a betting slip follows a specific lifecycle, which is concluded after all of its contained bets have been settled, i.e., all of its underlying events are over, and any realised winnings are either paid or credited to the player's account. A number of intermediate steps exist between these starting and ending points, and these steps may vary from one bookmaker system to another. The BMRS model for the betting slip lifecycle is depicted in Figure 2.2. Any external system sending data to the BMRS should be adjusted to follow this cycle as closely as possible by mapping its own statuses of a betting slip to the ones shown in the figure, which are further explained in Table 3.18.



**Figure 2.2:** The BMRS Betslip Lifecycle.

### 3.0 The BMRS API Processes

Any data transmission by the regulated entities to the NBA via the BMRS takes place using a number of API Methods in the form of SOAP requests (the communication protocols are described in detail in Section 4.0). Encoded in these methods are the data models the BMRS employs in structuring the various objects that collectively describe the processes and activities it is intended to monitor.

These activities may be grouped based on the objects they act on:

- Player Accounts
  - Create and Update Accounts.
  - Create Account Transactions (Deposits, Withdrawals etc.).
  - Monitor the addition, removal or expiration of Account Restrictions.
- Betting Activity
  - Create Bet Slips.
  - Update Bet Slips throughout their lifecycle.
- BMRS System-Related Data
  - Heartbeat call.
  - ErrorsLog.

The BMRS expects to receive data no more than **90-seconds** after it has been registered in the bookmaker's system. The API methods allow for the dispatch of such data in batches, meaning that multiple objects can be created or updated within a single call. Nevertheless, the choice for the actual process and scheduling followed is up to the sender and is constrained only by the 90- second limit.

The data structure of a BMRS method takes the form of an XML-tree. Some elements of that tree are simply containers under which a collection of one or more same-type sub-elements can be listed, while others contain sub-elements that represent data.

Common to all methods is the *<header>* element, which contains basic information that identify the sender of the data as shown in Table 3.1. The parameters under *<header>* are provided by the NBA prior to initiating a connection between an external system to the BMRS. Any subsequent changes to these identifiers are communicated by the Authority in good time.

Header				
No.	Field Name	Data Type	Description	Comment
1	DataEntryKey	String	A unique key generated by the BMRS and provided to each Bookmaker by the NBA (GUID).	This value can be replaced (re-issued) if needed.
2	LicenseNumber	String	The License Number of the Bookmaker.	
3	LicenseIdentifier	String	A Unique ID generated by the BMRS and provided to each Bookmaker by the NBA.	This value is <u>static</u> . Once issued it cannot change as it uniquely identifies the Bookmaker.

**Table 3.1:** Data fields (sub-elements) under the header element.

## 3.1 Player Accounts

The majority of processes monitored by the BMRS either originate from a Player Account (e.g. the placement of a Bet) or target a Player Account (e.g. a transaction resulting from winning a bet). As such, a Player's Account is central to the data model employed by the system and needs to be referenced every time an action involving that account is called. In terms of data flow, the

consequence of the above is that the creation of a Player's Account needs to precede any reporting that involves that account.

Player Accounts are created and may be updated post their creation by the same method, *SaveAccounts*.

**Note:** Accounts are required both for Class A and Class B operators, but the information they carry in the former case refer to shop cashiers, not to bettors.

### 3.1.1 Method *SaveAccounts*

#### 3.1.1.1 Description

Player Accounts created or updated in the Bookmaker's System at any time, need to be communicated to the BMRS within a 90-second time-window using the *SaveAccounts* method. The parameters that are passed on through this call follow the XML-type tree structure shown in Figure 3.1.

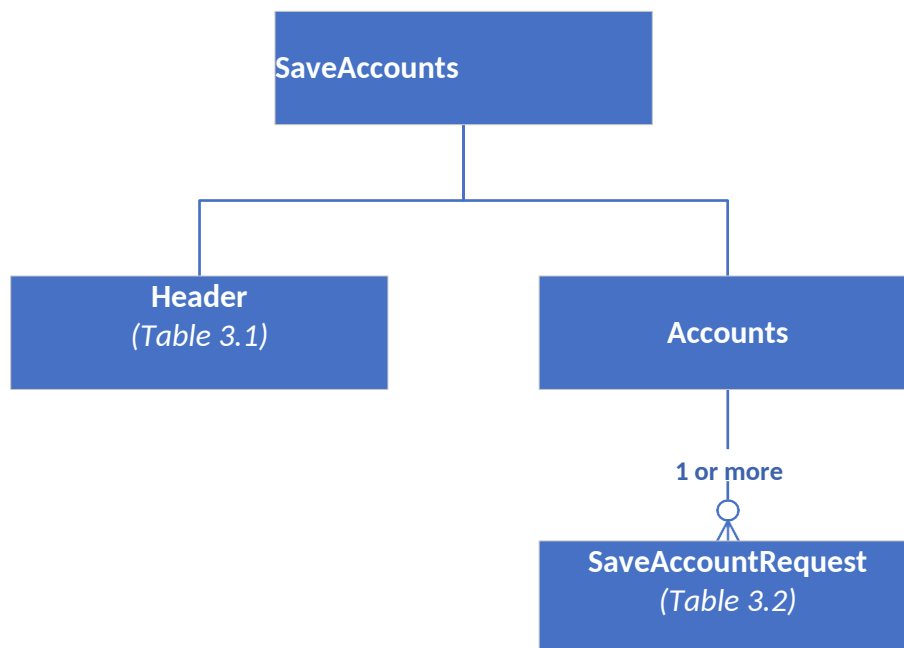
The *header* element and its contents have already been visited in Table 3.1.

The *SaveAccountRequest* node contains sub-elements that represent data fields related to the Account; these fields are listed in Table 3.2. An account may be designated as Active or Disabled, as described in Table 3.3.

**Note:** The **username** data field for an account needs to be a unique identifier. It is used in subsequent calls to reference that account (e.g. *CreateBetSlips*, *CreateAccountTransactions* etc.).

Except for the **username** (which is a unique identifier) and the **RegisteredOnDate** parameters, all other fields passed to the *SaveAccounts* method can be altered post the creation of a Player's Account. Thus, if at any point in time a player modifies some of this information in the Bookmaker's system, the changes should be reflected in the subsequent call to the BMRS API by including the complete set of their account details in a *SaveAccountRequest* element under the Accounts node.

### 3.1.1.2 Method Request



**Figure 3.1:** XML structure of the *SaveAccounts* method.

SSaveAccountRequest				
No.	Field Name	Data Type	Description	Comment
1	Username	String	The username of the Account. The <b>unique identifier</b> of this Account.	This needs not necessarily be an actual username, e.g. it could be a reference number.
2	RegisteredOnDate	DateTime	The UTC date and time this Account was created in the Bookmaker's system.	This set when the account is created and cannot be updated by subsequent calls.
3	FullName	String	Name and Surname of the Account owner.	
4	IdentityDocumentType	String	The type of the Identification Document of the Account Owner (National ID or Passport)	
5	IdentityNumber	String	The Identity Number of the Account owner (as per Identity Document Type selected)	
6	IdentificationDocumentIssuingCountry	String	The code of the identification document (National ID or Passport) issuing country.	<b>ISO-Alpha-2 or ISO-Alpha-3.</b>
7	DateOfBirth	DateTime	The Date of Birth of the Account owner.	
8	Status	Enumeration [Active] [Disabled]	The current Account status.	See Table 3.3 for more details.
9	IsVerified	Boolean	A flag indicating whether the identity/details of the Account owner have been verified by the Bookmaker.	
10	Description	String	Any additional information.	

**Table 3.2:** Data fields (sub-elements) under the SaveAccountRequest element.

Status	Description
Active	The Account is open to transactions (Deposit, Withdrawal, etc.) and/or can be involved in betting activity.
Disabled	The Account cannot be used for transactions or betting.

**Table 3.3:** *SaveAccountRequest - Status field values.*

**Note:** Multiple *SaveAccountRequest* nodes may be included in a single call to the *SaveAccounts* method, including both new Accounts and existing Accounts that have been updated. An account should only be included in the batch once; multiple occurrences of the same username within the same call would return an error.

An example of the SOAP/XML request for this method is shown below:

```

SaveAccount Sample Request
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/"
xmlns:int="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Request"
>
  <soapenv:Header/>
  <soapenv:Body>
    <tem:SaveAccounts>
      <tem:header>
        <int:DataEntryKey>F2D64583-DFC1-4DF1-A95D-02BCD1CBA489</int:DataEntryKey>
        <int:LicenseNumber>A-UT-0001</int:LicenseNumber>
        <int:LicenseeIdentifier>19666639-C7F8-4942-876D-CA1088B75EFE</int:LicenseeIdentifier>
      </tem:header>
      <tem:accounts>
        <!--One or more repetitions:-->
        <int:SaveAccountRequest>
          <int:DateOfBirth>1999-09-27T19:07:15.4993989+03:00</int:DateOfBirth>
          <int:Description>DESC_SPNTOSPIRG</int:Description>
          <int:FullName>First Last</int:FullName>
          <int:IsVerified>1</int:IsVerified>
          <int:IdentityDocumentType>NationalIdentityCard</int:IdentityDocumentType>
          <int:IdentityNumberIssuingCountry>CYP</int:IdentityNumberIssuingCountry>
          <int:IdentityNumber>00112233</int:IdentityNumber>
          <int:RegisteredOnDate>2018-09-27T19:07:15.5004083+03:00</int:RegisteredOnDate>
          <int>Status>Active</int>Status>
          <int:Username>UT-User_XBBQARPETW</int:Username>
        </int:SaveAccountRequest>
      </tem:accounts>
    </tem:SaveAccounts>
  </soapenv:Body>
</soapenv:Envelope>

```

### 3.1.1.3 Method Response

Following the call to any BMRS Method, the system provides feedback to the sender in the form of an XML response. In the case of creating and/or updating player accounts, this response includes feedback for each of the Accounts in the original call, collected under the *SaveAccountsResult* root element as shown in Figure 3.2. Note that each account is referenced by its assigned username.



**Figure 3.2:** XML Structure of the *SaveAccountsResponse*.

The types of possible errors and their description are listed in Table 3.4.

ErrorCode	ErrorMessage	Description
1500	"No Licensee Found for LicenseNumber:{0} LicenseIdentifier:{1} DataEntryKey:{2}"	Information about the licensee passed in the header do not match with any licensee in BMRS (E.g. Invalid LicenseNo, DataEntryKey etc.)
1504	"Field '{0}': value is null or empty"	A mandatory value or element is missing (e.g. the header of the request)
1505	"Field '{0}': value '{1}' is invalid"	The value passed is invalid (e.g. an invalid country code, undefined enumerations, non-expected negative or non-expected zero values)
1506	"Field '{0}': value '{1}' is duplicated"	Duplicate items exist in the same request (e.g. BetSlips, BetSlipItems, BetLines, Accounts, AccountTransactions, Restrictions)
1515	"Licensee {0} does not accept accounts. Licensee Type: {1}"	No account related requests are permitted unless the licensee is either Class A or B.

**Table 3.4:** Types of error in SaveAccountsResponse.

The following is an example of the SOAP/XML response:

SaveAccount Sample Response:
<p><b>Sample Response:</b></p> <pre> &lt;s:Envelope xmlns:stp"&gt; &lt;s:Body&gt; &lt;SaveAccountsResponse xmlns:itt"&gt; &lt;SaveAccountsResult xmlns:a=" nse" xmlns:itt"&gt; &lt;a:SaveAccountResponse&gt; &lt;a:ErrorCode&gt;0&lt;/a:ErrorCode&gt; &lt;a:ErrorMessage i:nil="true"/&gt; &lt;a:Success&gt;true&lt;/a:Success&gt; &lt;a:Username&gt;UT-User_XBBQARPETW&lt;/a:Username&gt; &lt;/a:SaveAccountResponse&gt; &lt;/SaveAccountsResult&gt; &lt;/SaveAccountsResponse&gt; &lt;/s:Body&gt; &lt;/s:Envelope&gt; </pre>

## 3.2 Account Transactions

### 3.2.1 Method CreateAccountTransactions

#### 3.2.1.1 Description

**Note:** Account Transactions apply only to Class B Licensees. No such data is expected from a Class A Bookmaker.

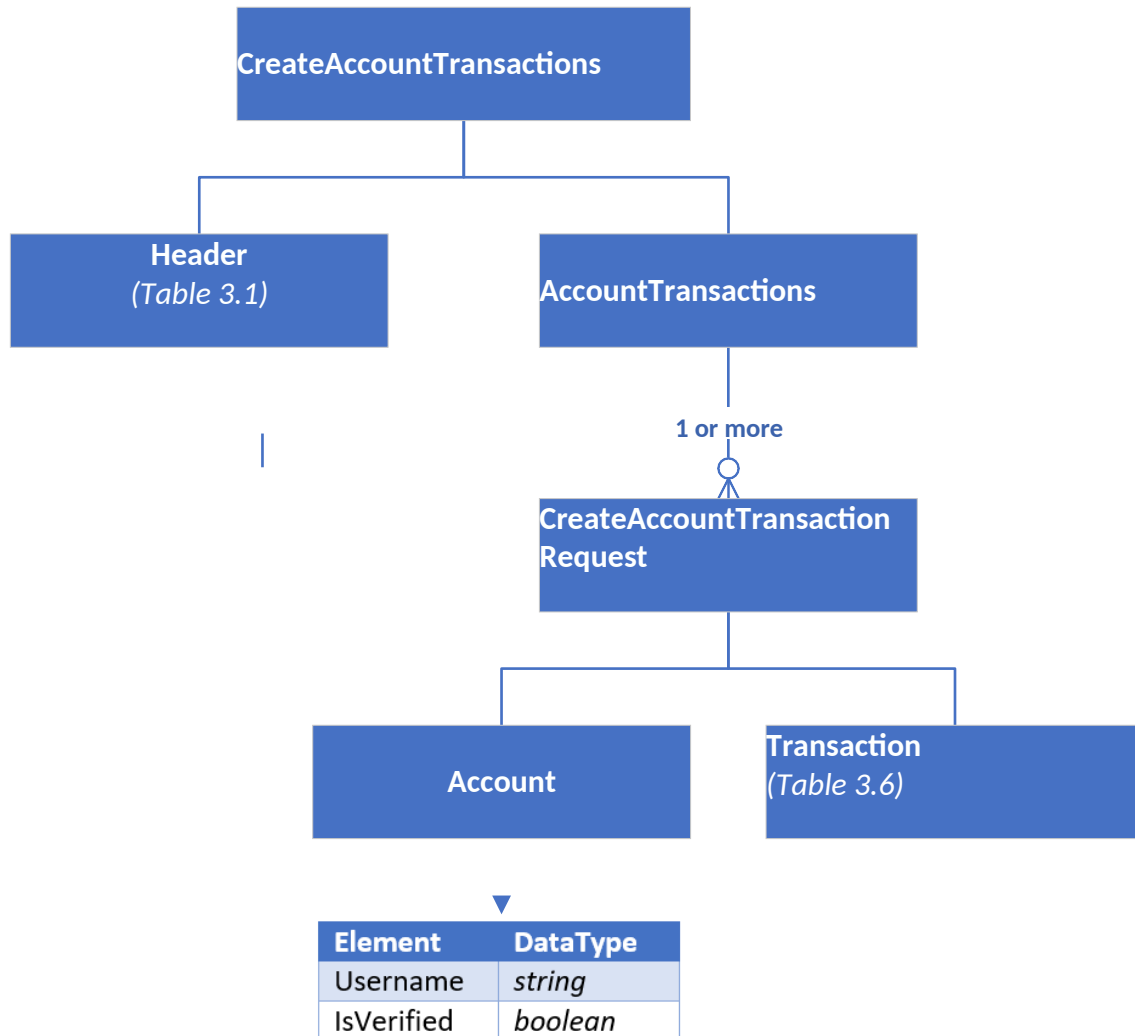
Player transactions reflect any transfer of funds in or out of a player’s account, the engagement of funds in wagering, and any conversion of bonus to real money. The full list of transaction types is shown in Table 3.5 below. Prior to a transaction of any kind being initiated in the BMRS, the account in question needs to have been created. Once sent, an account transaction cannot be updated.

TransactionType	Description	Required Fields
Deposit	Funds have been Deposited into the Account.	AmountMoney
Withdraw	Funds have been Withdrawn from the Account.	AmountMoney
BonusReal	An amount of Bonus has been converted to real money.	AmountBonus AND AmountMoney
BetWinnings	An amount (either real money or bonus or both) has been credited to the Account as a result of betting activities.	AmountMoney AND/OR AmountBonus
BetWager	An amount (either real money or bonus or both) has been committed in bet placement.	AmountMoney AND/OR AmountBonus
AddBonus	A bonus amount has been credited to the Account.	AmountBonus
RemoveBonus	A bonus amount has been taken out of the Account.	AmountBonus
Refund	An amount (either real money or bonus or both) have been refunded to the Account.	AmountMoney AND/OR AmountBonus

**Table 3.5:** CreateAccountTransactions - TransactionType field values.

### 3.2.1.2 Method Request

Any transactions of the types listed in Table 3.5 that have taken place in the bookmaker's platform need to be communicated to the Authority within a 90-second window. The *CreateAccountTransactions* method facilitates the transmission of this data in batches, where each batch may contain several such transactions (each with a different *TransactionReferenceNumber*) from multiple player accounts. The relevant data structure is shown in Figure 3.3. The elements under Transaction are described in Table 3.6.



**Figure 3.3:** The XML structure of the *CreateAccountTransactions* method.

Transaction				
No.	Field Name	Data Type	Description	Comment
1	TransactionReferenceNumber	String	A Unique Identifier to this transaction.	This is assigned by the Bookmaker's system.
2	TransactionType	Enumeration [Deposit] [Withdraw] [BonusReal] [BetWinnings] [BetWager] [AddBonus] [RemoveBonus] [Refund]	The type of Transaction.	See Table 3.6 for details.
3	AmountMoney	Decimal	The amount of real money involved in this Transaction.	
4	AmountBonus	Decimal	The amount of bonus involved in this Transaction.	
5	BalanceMoney	Decimal	The resultant Account Balance of real money <u>after</u> this Transaction.	This excludes any real money currently been wagered (field7).
6	BalanceBonus	Decimal	The resultant Account Balance of bonus <u>after</u> this Transaction.	This excludes any bonus currently been wagered (field8).
7	WageredMoney	Decimal	The total amount of real money from this Account that is wagered on outstanding bets <u>after</u> this Transaction.	
8	WageredBonus	Decimal	The total amount of bonus from this Account that is wagered on outstanding bets <u>after</u> this Transaction.	
9	CreatedOnDate	DateTime	The UTC date and time this transaction was created at in the Bookmaker's system.	

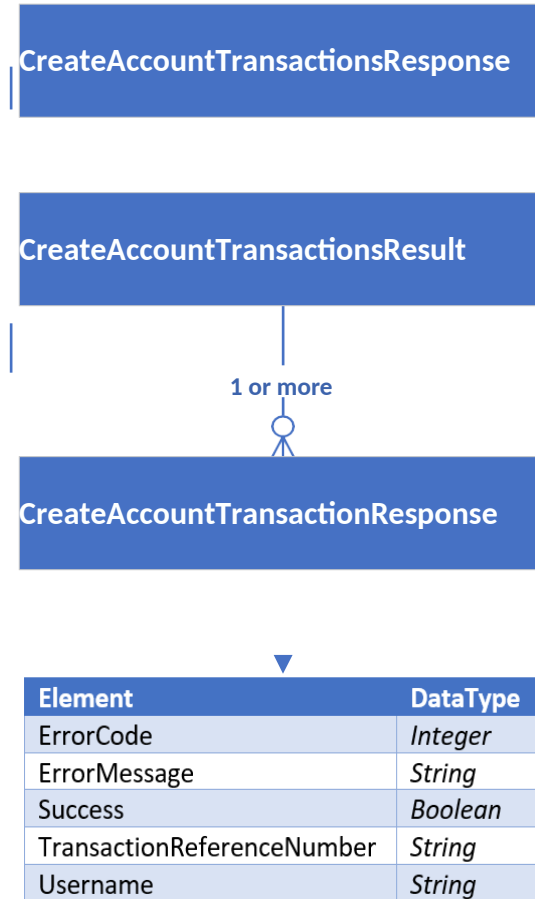
**Table 3.6:** Data fields (sub-elements) under the Transaction element.

An example of the SOAP/XML request for this method is shown below:

CreateAccountTransactions Sample Request
<pre> &lt;soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/" xmlns:int="http://schemas.datacontract.org/2004/07/IntelIScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Requ est"&gt;   &lt;soapenv:Header/&gt;   &lt;soapenv:Body&gt;     &lt;tem:CreateAccountTransactions&gt;       &lt;tem:header&gt;         &lt;int:DataEntryKey&gt;F2D64583-DFC1-4DF1-A95D-02BCD1CBA489&lt;/int:DataEntryKey&gt;         &lt;int:LicenseNumber&gt;A-UT-0001&lt;/int:LicenseNumber&gt;         &lt;int:LicenseeIdentifier&gt;19666639-C7F8-4942-876D-CA1088B75EFE&lt;/int:LicenseeIdentifier&gt;       &lt;/tem:header&gt;       &lt;tem:accountTransactions&gt;         &lt;!-- One or more repetitions:--&gt;         &lt;int:CreateAccountTransactionRequest&gt;           &lt;int:Account&gt;             &lt;int:IsVerified&gt;1&lt;/int:IsVerified&gt;             &lt;int:Username&gt;UT-User_QVNXIXOLGG&lt;/int:Username&gt;           &lt;/int:Account&gt;           &lt;int:Transaction&gt;             &lt;int:AmountBonus&gt;77&lt;/int:AmountBonus&gt;             &lt;int:AmountMoney&gt;2319&lt;/int:AmountMoney&gt;             &lt;int:BalanceBonus&gt;263&lt;/int:BalanceBonus&gt;             &lt;int:BalanceMoney&gt;4083&lt;/int:BalanceMoney&gt;             &lt;int:CreatedOnDate&gt;2019-09-27T19:11:54.3972355+03:00&lt;/int:CreatedOnDate&gt;             &lt;int:TransactionReferenceNumber&gt;TR-UT_KGLDDUNFMR_122312313&lt;/int:TransactionReferenceNumber&gt;             &lt;int:TransactionType&gt;BetWinnings&lt;/int:TransactionType&gt;             &lt;int:WageredBonus&gt;60&lt;/int:WageredBonus&gt;             &lt;int:WageredMoney&gt;53&lt;/int:WageredMoney&gt;           &lt;/int:Transaction&gt;         &lt;/int:CreateAccountTransactionRequest&gt;       &lt;/tem:accountTransactions&gt;     &lt;/tem:CreateAccountTransactions&gt;   &lt;/soapenv:Body&gt; &lt;/soapenv:Envelope&gt; </pre>

### 3.2.1.3 Method Response

The BMRS responds to a *CreateAccountTransactions* call with the XML structure shown in Figure 3.4. The response provides feedback on each and every transaction received through the API, by means of their *TransactionReferenceNumber* field. The types of error that might be returned are described in Table 3.7.



**Figure 3.4:** The XML structure of *CreateAccountTransactionsResponse*.

ErrorCode	ErrorMessage	Description
1500	"No Licensee Found for LicenseNumber:'{0}' LicenseeIdentifier:'{1}' DataEntryKey:'{2}'"	Information about the licensee passed in the header do not match with any licensee in BMRS (E.g. Invalid LicenseNo, DataEntryKey etc.)
1504	"Field '{0}': value is null or empty"	A mandatory value or element is missing (e.g. the header of the request)
1505	"Field '{0}': value '{1}' is invalid"	The value passed is invalid (e.g. an invalid country code, undefined enumerations, non-expected negative or non-expected zero values)
1506	"Field '{0}': value '{1}' is duplicated"	Duplicate items exist in the same request (e.g. BetSlips, BetSlipItems, BetLines, Accounts, AccountTransactions, Restrictions)
1515	"Licensee {0} does not accept accounts. Licensee Type: {1}"	No account related requests are permitted unless the licensee is either Class A or B.

**Table 3.7:** Types of error in *CreateAccountTransactionsResponse*.

The following is an example of the SOAP/XML response returned by this method:

CreateAccountTransactions Sample Response
<pre> &lt;s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"&gt;   &lt;s:Body&gt;     &lt;CreateAccountTransactionsResponse xmlns="http://tempuri.org/"&gt;       &lt;CreateAccountTransactionsResult xmlns:a="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Respo nse" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"&gt;         &lt;a:CreateAccountTransactionResponse&gt;           &lt;a:ErrorCode&gt;0&lt;/a:ErrorCode&gt;           &lt;a:ErrorMessage i:nil="true"/&gt;           &lt;a:Success&gt;true&lt;/a:Success&gt;           &lt;a:TransactionReferenceNumber&gt;TR-UT_KGLDDUNFMR_122312313&lt;/a:TransactionReferenceNumber&gt;           &lt;a:Username&gt;UT-User_QVNXIXOLGG&lt;/a:Username&gt;         &lt;/a:CreateAccountTransactionResponse&gt;       &lt;/CreateAccountTransactionsResult&gt;     &lt;/CreateAccountTransactionsResponse&gt;   &lt;/s:Body&gt; &lt;/s:Envelope&gt; </pre>

### 3.3 Account Restrictions

**Note:** Account Restrictions apply only to Class B Licensees. No such data is expected from a Class A Bookmaker.

Account Restrictions place limits on certain transactions or betting activities allowed by a player's account and these limits could be either self-imposed or applied by the bookmaker. The activation, de-activation and, in case there is a time dimension involved, the expiration of a restriction are the three events monitored by the BMRS, and if any of them happen to occur at any point in time, a call to the *SaveAccountRestrictions* function should be triggered within the next 90-seconds. The types of available Restrictions within the BMRS are described in Table 3.8.

RestrictionType	Description
ShortTermSelfExclusion	The Player have excluded themselves from placing any bet for a certain period of time.
LongTermSelfExclusion	The Player have excluded themselves from placing any bet indefinitely.
BettingAmountLimit	A Limit is placed on the total amount of money allowable for betting activities over a certain period by this Account.
BettingLossLimit	A Limit is placed on the total amount of money the Account may lose over a certain period.
DepositLimit	A limit is placed on the amount of money that can be deposited into this Account over a certain period.
Other	Any other restriction type not covered by the above. Details may be provided through the Description field (see Table 3.9).

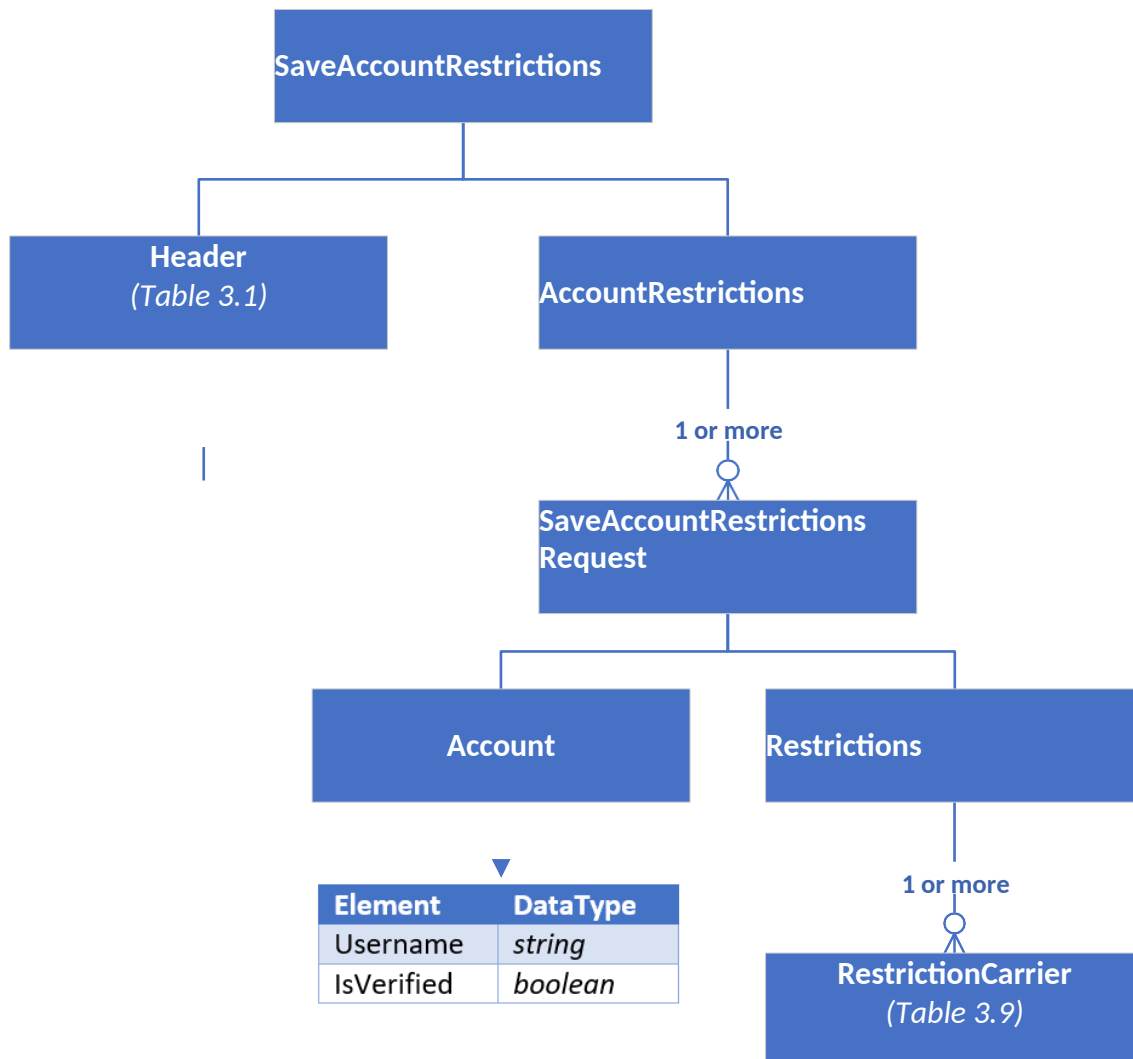
**Table 3.8:** *SaveAccountRestrictions* - **RestrictionType** field values.

### 3.3.1 Method SaveAccountRestrictions

#### 3.3.1.1 Description

The XML-structure adopted by this method is shown in Figure 3.5. The single data elements positioned under the *RestrictionCarrier* node of the tree are described separately in Table 3.9. As with all BMRS methods, a single call to *SaveAccountRestrictions* can accommodate batches of data, i.e. in this case multiple restriction-related events for one or more accounts.

#### 3.3.1.2 Method Request



**Figure 3.5:** The XML structure of the *SaveAccountRestrictions* method.

RestrictionCarrier				
No.	Field Name	Data Type	Description	Comment
1	EventName	Enumeration [Activated] [Deactivated] [Expired]	The event by which this Restriction was triggered.	
2	RestrictionType	Enumeration [ShortTermSelfExclusion] [LongTermSelfExclusion] [BettingAmountLimit] [BettingLossLimit] [DepositLimit] [Other]	The type of Restriction.	See Table 3.8 for details.
3	CreatedBy	Enumeration [Client] [Bookmaker]	Indicates whether this Restriction was imposed by the Player or the Bookmaker.	
4	CreatedOnDate	DateTime	UTC Date and Time the triggering event (field 1) occurred.	
5	Rationale	String	The reason behind this restriction.	Mainly for cases where the Restriction is imposed by the Bookmaker.
6	Description	String	Any additional information.	E.g. if RestrictionType is Other, this field may contain a brief description of what is limited by this Restriction.

**Table 3.9:** Data fields (sub-elements) under the RestrictionCarrier element.

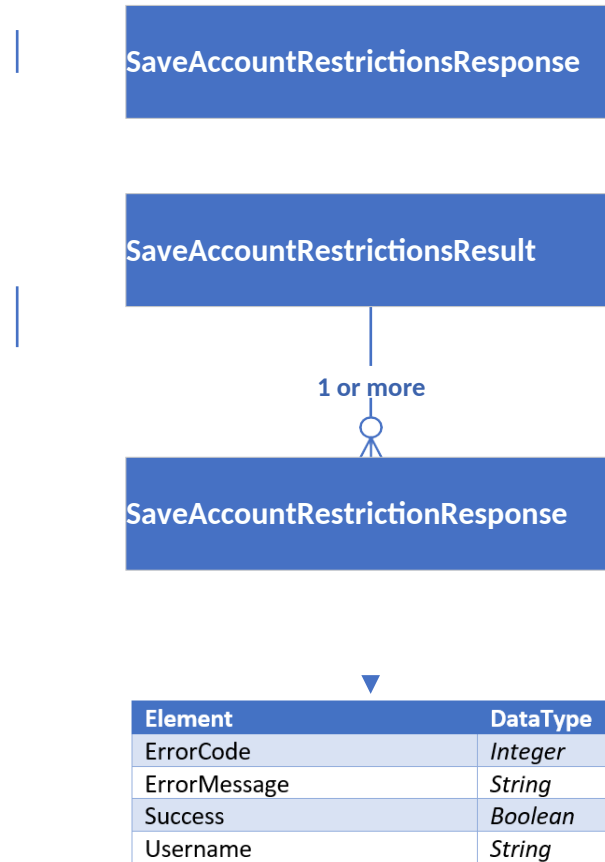
The following is an example of this methods SOAP/XML request:

**SaveAccountRestrictions Sample Request**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/"
xmlns:int="http://schemas.datacontract.org/2004/07/IntelIScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Requ
est">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:SaveAccountRestrictions>
      <tem:header>
        <int:DataEntryKey>F2D64583-DFC1-4DF1-A95D-02BCD1CBA489</int:DataEntryKey>
        <int:LicenseNumber>A-JT-0001</int:LicenseNumber>
        <int:LicenseeIdentifier>19666639-C7F8-4942-876D-CA1088B75EFE</int:LicenseeIdentifier>
      </tem:header>
      <tem:accountRestrictions>
        <!--One or more repetitions:-->
        <int:SaveAccountRestrictionRequest>
          <int:Account>
            <int:IsVerified>true</int:IsVerified>
            <int:Username>UT-User_CFMHOFYMVN</int:Username>
          </int:Account>
          <int:Restrictions>
            <!--One or more repetitions:-->
            <int:RestrictionCarrier>
              <int:CreatedBy>Bookmaker</int:CreatedBy>
              <int:CreatedOnDate>2019-09-27T18:59:56.3519795+03:00</int:CreatedOnDate>
              <int:Description>Description</int:Description>
              <int:EventName>Activated</int:EventName>
              <int:Rationale>Rationale</int:Rationale>
              <int:RestrictionType>BettingLossLimit</int:RestrictionType>
            </int:RestrictionCarrier>
          </int:Restrictions>
        </int:SaveAccountRestrictionRequest>
      </tem:accountRestrictions>
    </tem:SaveAccountRestrictions>
  </soapenv:Body>
</soapenv:Envelope>
```

### 3.3.1.3 Method Response

The BMRS responds to a *SaveAccountRestrictions* call with the XML structure shown in Figure 3.6. The response provides feedback on each restriction event received through the API, and these restrictions are loosely referenced by the username of the affected account. The types of error that might be returned are described in Table 3.10.



**Figure 3.6:** The XML structure of *SaveAccountRestrictionsResponse*.

ErrorCode	ErrorMessage	Description
1500	"No Licensee Found for LicenseNumber:'{0}' LicenseIdentifier:'{1}' DataEntryKey:'{2}'"	Information about the licensee passed in the header do not match with any licensee in BMRS (E.g. Invalid LicenseNo, DataEntryKey etc.)
1504	"Field '{0}': value is null or empty"	A mandatory value or element is missing (e.g. the header of the request)
1505	"Field '{0}': value '{1}' is invalid"	The value passed is invalid (e.g. an invalid country code, undefined enumerations, non-expected negative or non-expected zero values)
1506	"Field '{0}': value '{1}' is duplicated"	Duplicate items exist in the same request (e.g. BetSlips, BetSlipItems, BetLines, Accounts, AccountTransactions, Restrictions)
1515	"Licensee {0} does not accept accounts. Licensee Type: {1}"	No account related requests are permitted unless the licensee is either Class A or B.

**Table 3.10:** Types of error in SaveAccountRestrictionsResponse.

The following is an example of SOAP/XML response for this method:

SaveAccountRestrictions Sample Response:
<pre> &lt;s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"&gt;   &lt;s:Body&gt;     &lt;SaveAccountRestrictionsResponse xmlns="http://tempuri.org/"&gt;       &lt;SaveAccountRestrictionsResult xmlns:a="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Respo nse" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"&gt;         &lt;a:SaveAccountRestrictionResponse&gt;           &lt;a:ErrorCode&gt;0&lt;/a:ErrorCode&gt;           &lt;a:ErrorMessage i:nil="true"/&gt;           &lt;a:Success&gt;true&lt;/a:Success&gt;           &lt;a:Username&gt;UT-User_CFMHOFYMVN&lt;/a:Username&gt;         &lt;/a:SaveAccountRestrictionResponse&gt;       &lt;/SaveAccountRestrictionsResult&gt;     &lt;/SaveAccountRestrictionsResponse&gt;   &lt;/s:Body&gt; &lt;/s:Envelope&gt; </pre>

### 3.4 Betting Activity

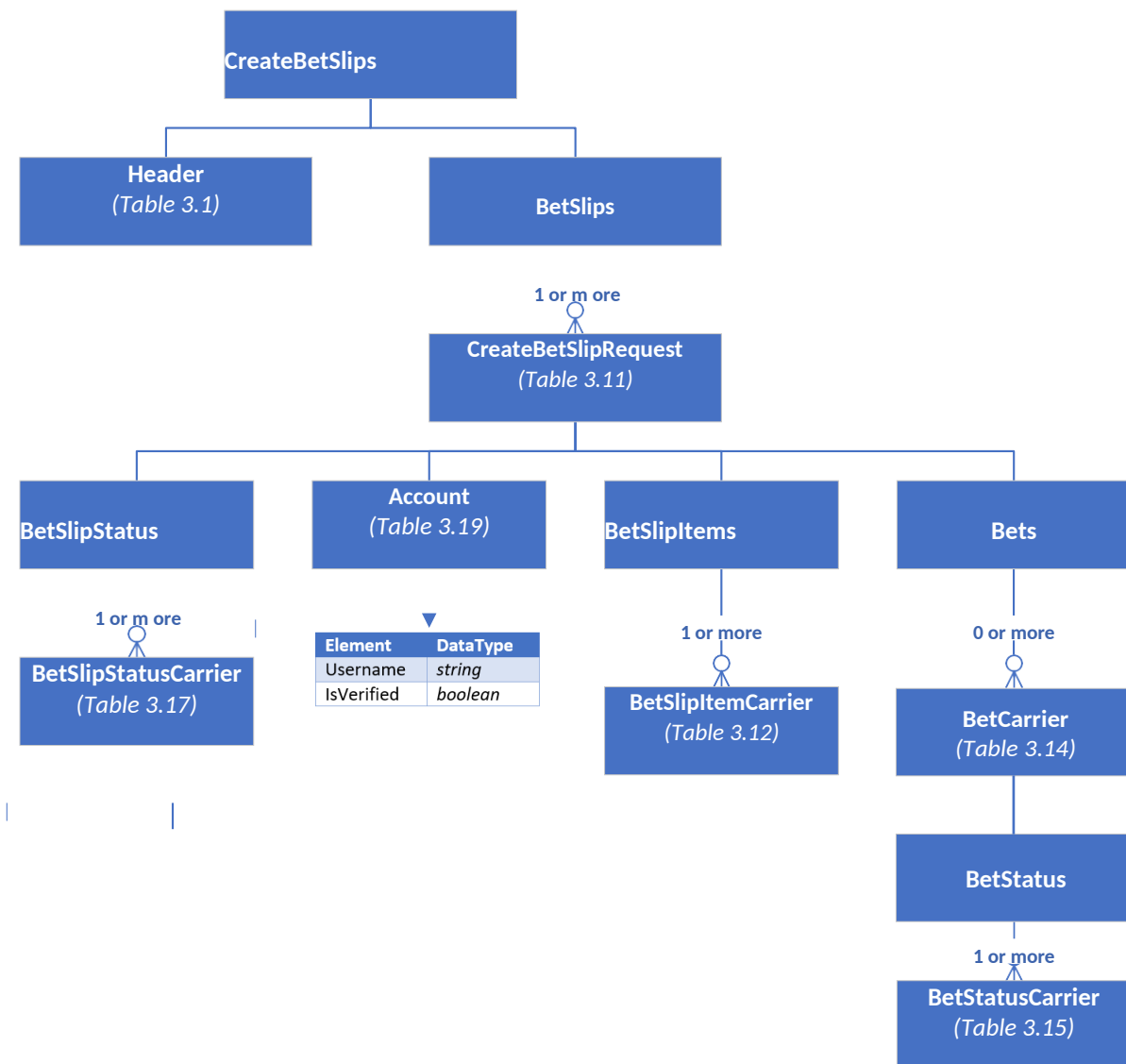
Betting activity within the context of the BMRS includes the placement of bets by player accounts, a detailed description of those bets and the tracking of their lifecycle through their status. Central to the data model employed for receiving betting data is the *BetSlip* object, which contains a variety of information, including multiple outcomes a player wishes to bet on and the actual *Bets* on those outcomes. An overview of the relationship between these objects is provided in Section 2.2.

### 3.4.1 Method CreateBetSlips

#### 3.4.1.1 Description

New bet slips created by one or multiple accounts can be communicated to the BMRS in batches by calling the *CreateBetSlips* method. The relevant XML structure is shown in Figure 3.7. The various data elements and the values they may take are described in Tables 3.11 to 3.19.

#### 3.4.1.2 Method Request



**Figure 3.7:** The XML structure of the *CreateBetSlips* method.

CreateBetSlipRequest				
No.	Field Name	Data Type	Description	Comment
1	ReferenceNumber	String	Unique Identifier assigned by the Bookmaker's system. No two BetSlips can have the same value.	Unique Identifier. Used to reference bet slip in consequent update requests.
2	IssuerLicenseNumber	String	The License Number of the issuer.	<u>Class A</u> : Shop License <u>Class B</u> : Bookmaker License
3	InitialStake	Decimal	The total initial stake on this BetSlip.	<u>Includes Bonus</u> .
4	InitialStakeBonus	Decimal	Any amount of Bonus on the initial stake.	
5	Payout	Decimal	The total final payout of this BetSlip after settlement of all Bets.	<u>Includes Bonus</u> .
6	PayoutBonus	Decimal	Any amount of bonus in the total final payout of this BetSlip.	
7	MaxPayout	Decimal	The maximum possible total Payout.	
8	MinPayout	Decimal	The minimum possible total Payout.	
9	Description	String	Any additional information.	
10	Commission	Decimal	Absolute amount of Commission paid to the Authorized Representative.	
11	TerminalId	String	Identifier of the terminal machine used to create this BetSlip.	Class A Only.
12	TotalNumberOfCombinations	Integer	Total number of combinations resulting from the underlying Bets.	
13	CreatedOnDate	DateTime	UTC Date and time of BetSlip creation in the Bookmaker's system.	

**Table 3.11:** Data fields (sub-elements) under the CreateBetSlipRequest element.

BetSliptemCarrier				
No.	Field Name	Data Type	Description	Comment
1	ItemReferenceNumber	String	Reference Number assigned to this BetSliptem by the Bookmaker's system.	Unique Identifier in this Betslip only.
2	BetType	Enumeration [Outright] [Pregame] [Live] [Other]	The bet type of this BetSliptem.	
3	Sport	String	The name of the sport of the underlying event.	
4	Competition	String	The name of the competition within which the underlying event takes place.	
5	Region	String	The geographic region over which the competition takes place.	
6	EventName	String	The name of the underlying event.	
7	EventStartDate	DateTime	The UTC date and time at which the underlying event starts.	
8	EventResult	String	The final result of the underlying event.	
9	EventKey	String	The unique identifier assigned to the underlying event by the Bookmaker's system.	
10	CompetitorA	String	The name of Competitor A.	If Applicable.
11	CompetitorB	String	The name of Competitor B.	If Applicable.
12	MarketType	String	The name of the Market Type.	Examples: "Full Time Result" "Under/Over 2.5"
13	MarketTypeKey	String	The unique identifier assigned to this Market Type by the Bookmaker's system.	
14	MarketTypeKeyBMRS	String	The unique identifier assigned to this Market Type by the BMRS.	Supplied by the NBA.
15	SelectionName	String	The name of the player's selection.	Examples: "Home To Win" "Over"
16	SelectionKeyBMRS	String	The unique identifier assigned to the player's selection by the BMRS.	Supplied by the NBA.
17	Odds	Decimal	The odds offered by the Bookmaker for the player's selection.	
18	Description	String	Any additional information.	

19	Status	Enumeration: [Pending] [Won] [Lost] [Void]	The Status of the player's selection for the market of this item.	See Table 3.13.
----	--------	--	---	-----------------

**Table 3.12:** Data fields (sub-elements) under the BetSliptemCarrier element.

Status	Description
Pending	The outcome of the market has not been decided yet.
Won	The outcome of the market is the same as the player's selection.
Lost	The outcome of the market is different to the player's selection.
Void	The outcome of the market cannot be determined (e.g. event cancelled).

**Table 3.13:** BetSliptemCarrier - Status field values.

BetCarrier				
No.	Field Name	Data Type	Description	Comment
1	BetReferenceNumber	String	Reference number of this Bet. Assigned by the Bookmaker's system.	Unique identifier within this BetSlip only.
2	BetSliptemReferenceNumbers	String	Comma-separated string of the Reference Number (Table 3.12 Field 1) of each BetSliptem included in this Bet.	
3	Description	String	Any additional information.	
4	NumberOfCombinations	Integer	The total number of combinations derived from this Bet.	
5	MinOdds	Decimal	Minimum odds offered for this Bet.	
6	MaxOdds	Decimal	Maximum odds offered for this Bet.	
7	InitialStake	Decimal	The total initial stake on this Bet. Includes Bonus.	

8	Payout	Decimal	The total final payout of this Bet. Includes Bonus.	
---	--------	---------	--	--

**Table 3.14:** Data fields (sub-elements) under the BetCarrier element.

BetStatusCarrier				
No.	Field Name	Data Type	Description	Comment
1	Status	Enumeration [Pending] [Won] [Lost] [Void]	The current status of this Bet.	See Table 3.16.
2	CreatedOnDate	DateTime	UTC Date and time this status was attained in the Bookmaker's system.	

**Table 3.15:** Data fields (sub-elements) under the BetStatusCarrier element.

Status	Description
Pending	The Bet is still pending.
Won	At least one of the underlying combinations of the Bet was realised.
Lost	None of the underlying combinations of the Bet was realised.
Void	The bet is no longer considered valid.

**Table 3.16:** BetStatusCarrier - Status field values.

BetSlipStatusCarrier				
No.	Field Name	Data Type	Description	Comment
1	Status	Enumeration [Submitted] [Accepted] [Cancelled] [Rejected] [Won] [Lost] [Refund] [PaidWon] [PaidRefund]	The current status of this BetSlip.	See Figure 2.2 and Table 3.18 for the bet slip lifecycle.
2	CurrentPayout	Decimal	Total Payout already secured by this BetSlip. Includes bonus.	
3	SettledStake	Decimal	Any amount of stake that has been settled. Includes bonus.	
4	CreatedOnDate	DateTime	The UTC Date and Time when this change in Status or CurrentPayout or SettledStake was recorded in the Bookmaker's system.	

**Table 3.17:** Data fields (sub-elements) under the BetSlipStatusCarrier element.

Status	Description
Submitted	The BetSlip has been created in the Bookmaker's system.
Accepted	The Bookmaker has accepted the BetSlip.
Cancelled	The Player has cancelled the betting slip.
Rejected	The Bookmaker has rejected the BetSlip.
Won	One or more Bets on the BetSlip have won. The amount has not been paid to the Player yet.
Lost	None of the Bets on the BetSlip have won.
Refund	The stake will be refunded to the Player.
PaidWon	The winnings have been paid to the Player.
PaidRefund	The refund has been paid to the Player.

**Table 3.18:** BetSlipStatusCarrier - Status field values.

**Note:** Each betting slip created by passing data to the *CreateBetSlips* method can have multiple *BetSlipStatusCarrier* and/or *BetStatusCarrier* elements attached to it (Figure 3.7). This feature allows the inclusion of more than one changes in the underlying parameters of these nodes, within the same call. As an example, if a newly created betslip has been Submitted and Accepted within the same 90-second window, the sender may opt to transmit both changes in a single call to this method by including two distinct and suitably time- stamped *BetSlipStatusCarrier* elements to its XML structure.

Account				
No.	Field Name	Data Type	Description	Comment
1	Username	String	Username (UID) of the Account that placed this Bet.	For Class A this will be a cashier.
2	IsVerified	Boolean	Flag for whether the Account that placed this BetSlip is a verified account or not.	

**Table 3.19:** Data fields (sub-elements) under the Account element.

Below is an example of SOAP/XML request structure for this method:

**CreateBetSlips Sample Request:**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/"
xmlns:int="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Requ
est">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:CreateBetSlips>
      <tem:header>
        <int:DataEntryKey>F2D64583-DFC1-4DF1-A95D-02BCD1CBA489</int:DataEntryKey>
        <int:LicenseNumber>A-UT-0001</int:LicenseNumber>
        <int:LicenseeIdentifier>19666639-C7F8-4942-876D-CA1088B75EFE</int:LicenseeIdentifier>
      </tem:header>
      <tem:betSlips>
        <!--One or more repetitions:-->
        <int:CreateBetSlipRequest>

          <int:BetSlipStatus>
            <!--One or more repetitions:-->
            <BetSlipStatusCarrier>
              <int:CreatedOnDate>2019-09-27T19:19:13.6183845+03:00</int:CreatedOnDate>
              <int:CurrentPayout>412</int:CurrentPayout>
              <int:SettledStake>26</int:SettledStake>
              <int:Status>Submitted</int:Status>
            </BetSlipStatusCarrier>
          </int:BetSlipStatus>
        </tem:betSlips>
      </tem:CreateBetSlips>
    </soapenv:Body>
  </soapenv:Envelope>
```

```

<int:CreatedOnDate>2019-09-27T19:20:13.6183845+03:00</int:CreatedOnDate>
<int:CurrentPayout>412</int:CurrentPayout>
<int:SettledStake>26</int:SettledStake>
<int:Status>Accepted</int:Status>
</BetSlipStatusCarrier>
</int:BetSlipStatus>

<int:Commision>1</int:Commision>
<int:Description>Unit Test APSOHAUYUI</int:Description>
<int:Payout>55</int:Payout>
<int:PayoutBonus>55</int:PayoutBonus>
<int:ReferenceNumber>UT-38115dcca4ce4-1277-4c56-b48b-7ac46018123</int:ReferenceNumber>
<int:IssuerLicenseNumber>SHOP-UT-0002</int:IssuerLicenseNumber>

<int:Account>
  <int:IsVerified>true</int:IsVerified>
  <int:Username>UT-User332014</int:Username>
</int:Account>

<int:BetSlipItems>
  <!-- One or more repetitions:-->
  <int:CreateBetSlipRequest.BetSlipItemCarrier>
    <int:EventResult>1</int:EventResult>
    <int:EventStartDate>2019-10-07T19:20:13.6193833+03:00</int:EventStartDate>
    <int:ItemReferenceNumber>1</int:ItemReferenceNumber>
    <int:Status>Pending</int:Status>
    <int:BetType>Outright</int:BetType>
    <int:Competition>UEFA Champions League</int:Competition>
    <int:CompetitorA>Team 01</int:CompetitorA>
    <int:CompetitorB>Team 02</int:CompetitorB>
    <int:Description>Unit Test PTEQMSOCPU</int:Description>
    <int:EventKey>1</int:EventKey>
    <int:EventName>Team 01 Vs Team 15</int:EventName>
    <int:MarketType>Correct Score</int:MarketType>
    <int:MarketTypeKey>Correct Score</int:MarketTypeKey>
    <int:MarketTypeKeyBMRS>Correct Score</int:MarketTypeKeyBMRS>
    <int:Odds>8</int:Odds>
    <int:Region>Europe</int:Region>
    <int:SelectionKeyBMRS>1</int:SelectionKeyBMRS>
    <int:SelectionName>1</int:SelectionName>
    <int:Sport>Soccer</int:Sport>
  </int:CreateBetSlipRequest.BetSlipItemCarrier>

  <int:CreateBetSlipRequest.BetSlipItemCarrier>
    <int:EventResult>1</int:EventResult>
    <int:EventStartDate>2019-10-07T19:20:13.621383+03:00</int:EventStartDate>
    <int:ItemReferenceNumber>2</int:ItemReferenceNumber>
    <int:Status>Pending</int:Status>
    <int:BetType>Outright</int:BetType>
    <int:Competition>UEFA Champions League</int:Competition>
    <int:CompetitorA>Team 01</int:CompetitorA>
    <int:CompetitorB>Team 02</int:CompetitorB>
    <int:Description>Unit Test YGPIJAEKNL</int:Description>
    <int:EventKey>1</int:EventKey>
    <int:EventName>Team 01 Vs Team 16</int:EventName>
    <int:MarketType>Correct Score</int:MarketType>
    <int:MarketTypeKey>Correct Score</int:MarketTypeKey>
    <int:MarketTypeKeyBMRS>Correct Score</int:MarketTypeKeyBMRS>
    <int:Odds>6</int:Odds>
    <int:Region>Europe</int:Region>
    <int:SelectionKeyBMRS>1</int:SelectionKeyBMRS>
    <int:SelectionName>1</int:SelectionName>
    <int:Sport>Soccer</int:Sport>
  </int:CreateBetSlipRequest.BetSlipItemCarrier>

</int:BetSlipItems>

<int:Bets>
  <!-- Zero or more repetitions:-->
  <int:CreateBetSlipRequest.BetCarrier>

```

```

<int:BetReferenceNumber>1</int:BetReferenceNumber>
<int:BetStatus>
  <!-- One or more repetitions:-->
  <int:BetStatusCarrier>
    <int:CreatedOnDate>2019-09-27T19:20:13.6223832+03:00</int:CreatedOnDate>
    <int:Status>Pending</int:Status>
  </int:BetStatusCarrier>
</int:BetStatus>
<int:Payout>100</int:Payout>
<int:BetSlipItemReferenceNumbers>1</int:BetSlipItemReferenceNumbers>
<int:Description>Unit Test GGVRFPBDS</int:Description>
<int:InitialStake>22</int:InitialStake>
<int:MaxOdds>2</int:MaxOdds>
<int:MinOdds>5</int:MinOdds>
<int:NumberOfCombinations>86</int:NumberOfCombinations>
</int:CreateBetSlipRequest.BetCarrier>

<int:CreateBetSlipRequest.BetCarrier>
  <int:BetReferenceNumber>2</int:BetReferenceNumber>
  <int:BetStatus>
    <!-- One or more repetitions:-->
    <int:BetStatusCarrier>
      <int:CreatedOnDate>2019-09-27T19:20:13.6233824+03:00</int:CreatedOnDate>
      <int:Status>Pending</int:Status>
    </int:BetStatusCarrier>
  </int:BetStatus>
  <int:Payout>100</int:Payout>
  <int:BetSlipItemReferenceNumbers>2</int:BetSlipItemReferenceNumbers>
  <int:Description>Unit Test ARMUBSNYNQ</int:Description>
  <int:InitialStake>67</int:InitialStake>
  <int:MaxOdds>1</int:MaxOdds>
  <int:MinOdds>9</int:MinOdds>
  <int:NumberOfCombinations>28</int:NumberOfCombinations>
</int:CreateBetSlipRequest.BetCarrier>

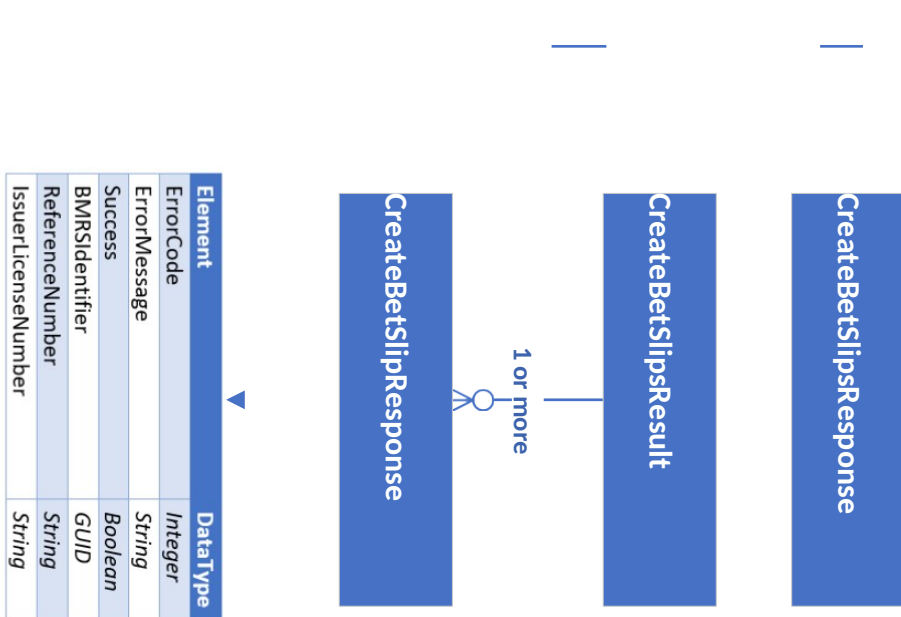
</int:Bets>

<int:CreatedOnDate>2019-09-27T19:20:13.6166518+03:00</int:CreatedOnDate>
<int:InitialStake>16</int:InitialStake>
<int:InitialStakeBonus>20</int:InitialStakeBonus>
<int:MaxPayout>100</int:MaxPayout>
<int:MinPayout>20</int:MinPayout>
<int:TerminalId>FLWVE</int:TerminalId>
<int:TotalNumberOfCombinations>2</int:TotalNumberOfCombinations>
</int:CreateBetSlipRequest>
</tem:betSlips>
</tem:CreateBetSlips>
</soapenv:Body>
</soapenv:Envelope>

```

### 3.4.2.3 Method Response

When the *CreateBetSlips* method is called, the API responds with the XML structure shown in Figure 3.8. The response provides feedback on each and every bet slip received, by means of its *ReferenceNumber* field. The types of error that might be returned are described in Table 3.20.



**Figure 3.8:** The XML structure of *CreateBetSlipsResponse*.

ErrorCode	ErrorMessage	Description
1500	"No Licensee Found for LicenseNumber:'{0}' LicenseeIdentifier:'{1}' DataEntryKey:'{2}'"	Information about the licensee passed in the header do not match with any licensee in BMRS (E.g. Invalid LicenseNo, DataEntryKey etc.).
1501	"Representative Licensee '{0}' not found"	Unknown representative license
1503	"Licensee {0} does not represents Licensee {1}"	Representative License does not represent bookmaker.
1504	"Field '{0}': value is null or empty"	A mandatory value or element is missing (e.g. the header of the request).
1505	"Field '{0}': value '{1}' is invalid"	The value passed is invalid (e.g. an invalid country codes, undefined enumerations, non-expected negative or non-expected zero values).
1506	"Field '{0}': value '{1}' is duplicated"	Duplicate items exist in the same request (e.g. BetSlips, BetSlipItems, BetLines, Accounts, AccountTransactions, Restrictions).
1514	"Licensee {0} does not accept bets. Licensee Type: {1}"	Betslips are allowed to be only for ClassB or SHOP. Class A and representatives must associate their betslips with a shop.

**Table 3.20:** Types of error in CreateBetSlipsResponse.

The following is an example of this methods SOAP/XML response:

CreateBetSlips Sample Response
<pre> &lt;s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"&gt;   &lt;s:Body&gt;     &lt;CreateBetSlipsResponse xmlns="http://tempuri.org/"&gt;       &lt;CreateBetSlipsResult xmlns:a="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Respo nse" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"&gt;         &lt;a:CreateBetSlipResponse&gt;           &lt;a:ErrorCode&gt;0&lt;/a:ErrorCode&gt;           &lt;a:ErrorMessage i:nil="true"/&gt;           &lt;a:Success&gt;true&lt;/a:Success&gt;           &lt;a:BMRSIdentifier&gt;f7971e7c-09e5-47a6-a980-38822fe35597&lt;/a:BMRSIdentifier&gt;           &lt;a:ReferenceNumber&gt;UT-38115dcca4ce4-1277-4c56-b48b-7ac46018123&lt;/a:ReferenceNumber&gt;           &lt;a:IssuerLicenseNumber&gt;SHOP-UT-0002&lt;/a:IssuerLicenseNumber&gt;         &lt;/a:CreateBetSlipResponse&gt;       &lt;/CreateBetSlipsResult&gt;     &lt;/CreateBetSlipsResponse&gt;   &lt;/s:Body&gt; &lt;/s:Envelope&gt; </pre>

### 3.4.3 Method UpdateBetSlips

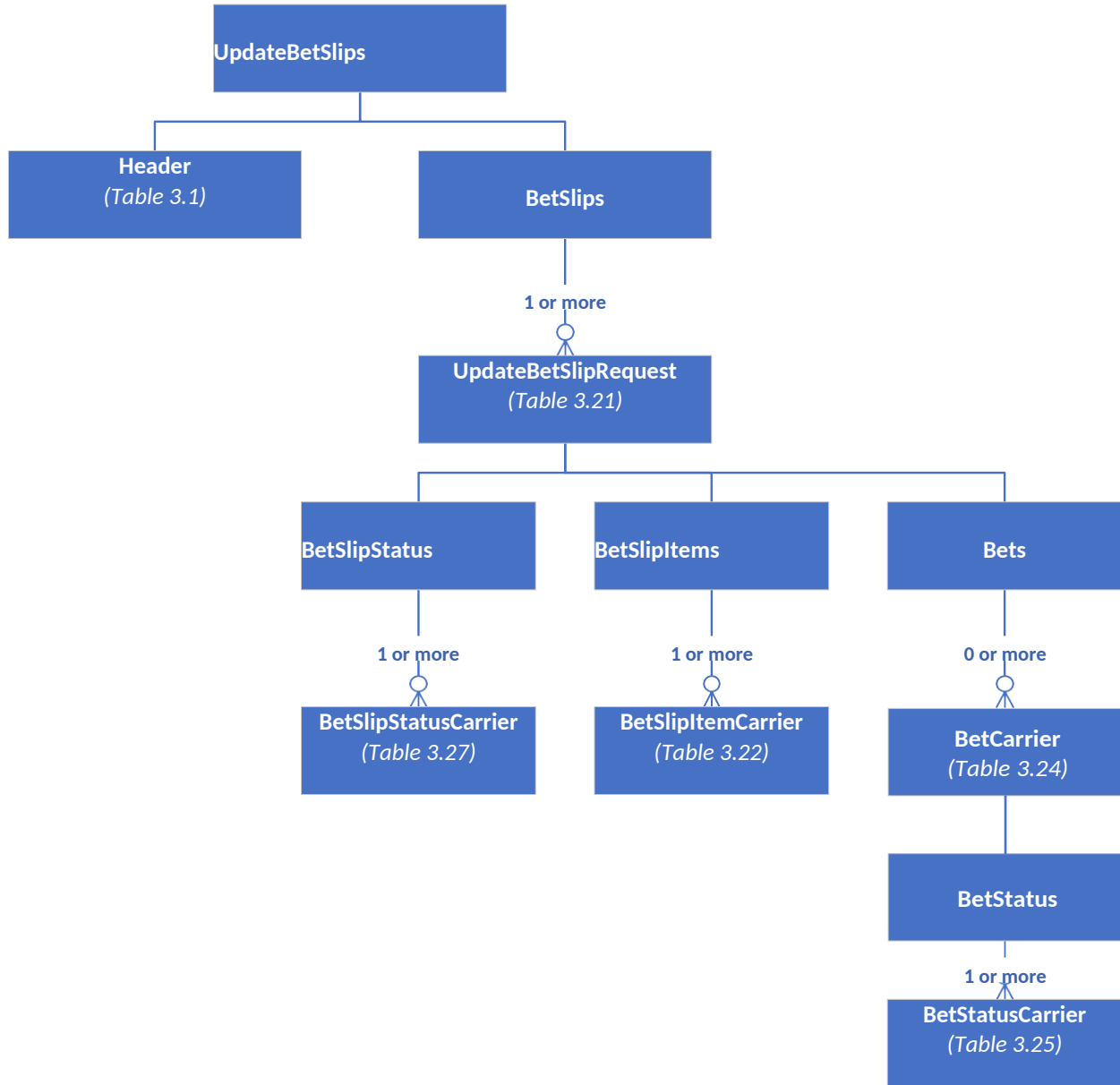
#### 3.4.3.1 Description

Once created, BetSlips follow their lifecycle towards settlement, upon which the player that created a betting slip has either lost or received their winnings (or refund in case the slip became void). A change in the BetSlip Status (table 3.18), the conclusion and outcome of a BetSlip item, a partial cash-out by a player or the change in status of a constituent Bet on the slip, are all examples where the state of the BetSlip has been altered.

The BMRS API provides the *UpdateBetSlips* method to reflect any such changes in the state of a betting slip. The method accepts data used to reference a betslip, alongside data for the fields that require an update. As with previously visited BMRS functions, *UpdateBetSlips* can take its data in batches, i.e. multiple slips can be updated via a single call. Each batch can only contain a bet slip *ReferenceNumber* once.

### 3.4.3.2 Method Request

The XML structure of this method is shown in Figure 3.9. Single data elements are described in Tables 3.21 to 3.28.



**Figure 3.9:** The XML structure of the UpdateBetSlips method.

UpdateBetSlipRequest				
No.	Field Name	Data Type	Description	Comment
1	ReferenceNumber	String	Unique Identifier assigned by the Bookmaker's system.	Used to reference the slip that is subject to this Update.
2	IssuerLicenseNumber	String	The License Number of the betslip issuer.	
3	Payout	Decimal	The total final payout of this BetSlip after settlement of all Bets.	Includes Bonus.
4	PayoutBonus	Decimal	Any amount of bonus in the total final payout of this BetSlip.	
5	Description	String	Any additional information.	
6	Commission	Decimal	Absolute amount of Commission paid to the Authorized Representative.	

**Table 3.21:** Data fields (sub-elements) under the UpdateBetSlipRequest element.

BetSlipItemCarrier				
No.	Field Name	Data Type	Description	Comment
1	ItemReferenceNumber	String	Reference Number assigned to this BetSlipItem by the Bookmaker's system.	Unique Identifier in this Betslip only. Used here to reference the Item subject to this Update action.
2	EventStartDate	DateTime	The UTC date and time at which the underlying event starts.	
3	EventResult	String	The final result of the underlying event.	
4	Status	Enumeration: [Pending] [Won] [Lost] [Void]	The Status of the player's selection for the market of this item.	See Table 3.23.

**Table 3.22:** Data fields (sub-elements) under the BetSlipItemCarrier element.

Status	Description
Pending	The outcome of the market has not been decided yet.
Won	The outcome of the market is the same as the player's selection.
Lost	The outcome of the market is different to the player's selection.
Void	The outcome of the market cannot be determined (e.g. event cancelled).

**Table 3.23:** *BetSlipItemCarrier* - **Status** field values.

BetCarrier				
No.	Field Name	Data Type	Description	Comment
1	BetReferenceNumber	String	Reference number of this Bet. Assigned by the Bookmaker's system.	Unique identifier within this BetSlip only. Used here to reference the Bet.
2	Payout	Decimal	The total final payout of this Bet. Includes bonus.	

**Table 3.24:** *Data fields (sub-elements) under the BetCarrier element.*

BetStatusCarrier				
No.	Field Name	Data Type	Description	Comment
1	Status	Enumeration [Pending] [Won] [Lost] [Void]	The current status of this Bet.	See Table 3.26.
2	CreatedOnDate	DateTime	UTC Date and time this status was attained in the Bookmaker's system.	

**Table 3.25:** *Data fields (sub-elements) under the BetStatusCarrier element.*

Status	Description
Pending	The Bet is still pending.
Won	At least one of the underlying combinations of the Bet was realised.
Lost	None of the underlying combinations of the Bet was realised.
Void	The bet is no longer considered valid.

**Table 3.26:** *BetStatusCarrier* - **Status** field values.

BetSlipStatusCarrier				
No.	Field Name	Data Type	Description	Comment
1	Status	Enumeration [Submitted] [Accepted] [Cancelled] [Rejected] [Won] [Lost] [Refund] [PaidWon] [PaidRefund]	The current status of this BetSlip.	See Figure 2.2 and table 3.28 for the bet slip lifecycle
2	CurrentPayout	Decimal	Any Payout secured by this BetSlip. Includes bonus.	
3	SettledStake	Decimal	Any amount of stake on this Betslip that has been settled. Includes bonus.	
4	CreatedOnDate	DateTime	The UTC Date and Time when this change in Status or CurrentPayout or SettledStake was recorded in the Bookmaker's system.	

**Table 3.27:** *Data fields (sub-elements) under the BetSlipStatusCarrier element.*

Status	Description
Submitted	The BetSlip has been created in the Bookmaker's system.
Accepted	The Bookmaker has accepted the BetSlip.
Cancelled	The Player has cancelled the betting slip.
Rejected	The Bookmaker has rejected the BetSlip.
Won	One or more Bets on the BetSlip have won. The amount has not been paid to the Player yet.
Lost	None of the Bets on the BetSlip have won.
Refund	The stake will be refunded to the Player.
PaidWon	The winnings have been paid to the Player.
PaidRefund	The refund has been paid to the Player.

**Table 3.28:** *BetSlipStatusCarrier - Status field values.*

**Note:** Each betting slip updated by passing data to the *UpdateBetSlips* method can have multiple *BetSlipStatusCarrier* and/or *BetStatusCarrier* elements attached to it (Figure 3.9). This feature allows the inclusion of more than one changes in the underlying parameters of these nodes, within the same call. As an example, if a betslip has been Won and PaidWon within the same 90-second window, the sender may opt to transmit both changes in a single call to this method by including two distinct and suitably time-stamped *BetSlipStatusCarrier* elements to its XML structure.

The following is an example SOAP/XML request for this method:

**UpdateBetSlips Sample Request:**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/"
xmlns:int="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Requ
est">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:UpdateBetSlips>

      <tem:header>
        <int:DataEntryKey>F2D64583-DFC1-4DF1-A95D-02BCD1CBA489</int:DataEntryKey>
        <int:LicenseNumber>A-UT-0001</int:LicenseNumber>
        <int:LicenseeIdentifier>19666639-C7F8-4942-876D-CA1088B75EFE</int:LicenseeIdentifier>
      </tem:header>

      <tem:betSlips>
        <!--One or more repetitions-->
        <int:UpdateBetSlipRequest>
```

```

<int:BetSlipStatus>
  <!--One or more repetitions:-->
  <BetSlipStatusCarrier>
    <int:CreatedOnDate>2019-09-27T19:20:12.6183845+03:00</int:CreatedOnDate>
    <int:CurrentPayout>412</int:CurrentPayout>
    <int:SettledStake>26</int:SettledStake>
    <int:Status>Won</int:Status>
  </BetSlipStatusCarrier>
  <BetSlipStatusCarrier>
    <int:CreatedOnDate>2019-09-27T19:20:13.6183845+03:00</int:CreatedOnDate>
    <int:CurrentPayout>412</int:CurrentPayout>
    <int:SettledStake>26</int:SettledStake>
    <int:Status>PaidWon</int:Status>
  </BetSlipStatusCarrier>
</int:BetSlipStatus>

<int:Commision>1</int:Commision>
<int:Description>description</int:Description>
<int:Payout>100</int:Payout>
<int:PayoutBonus>10</int:PayoutBonus>
<int:ReferenceNumber>UT-38115dcca4ce4-1277-4c56-b48b-7ac46018123</int:ReferenceNumber>
<int:IssuerLicenseNumber>SHOP-UT-0002</int:IssuerLicenseNumber>

<int:BetSlipItems>
  <!--One or more repetitions:-->
  <int:UpdateBetSlipRequest.BetSlipItemCarrier>
    <int:EventResult>1</int:EventResult>
    <int:EventStartDate>2019-10-07T19:20:13.6193833+03:00</int:EventStartDate>
    <int:ItemReferenceNumber>1</int:ItemReferenceNumber>
    <int:Status>Won</int:Status>
  </int:UpdateBetSlipRequest.BetSlipItemCarrier>

  <int:UpdateBetSlipRequest.BetSlipItemCarrier>
    <int:EventResult>2</int:EventResult>
    <int:EventStartDate>2019-10-07T19:20:13.621383+03:00</int:EventStartDate>
    <int:ItemReferenceNumber>2</int:ItemReferenceNumber>
    <int:Status>Won</int:Status>
  </int:UpdateBetSlipRequest.BetSlipItemCarrier>

</int:BetSlipItems>

<int:Bets>
  <!--Zero or more repetitions:-->
  <int:UpdateBetSlipRequest.BetCarrier>
    <int:BetReferenceNumber>1</int:BetReferenceNumber>
    <int:BetStatus>
      <!-- One or more repetitions:-->
      <int:BetStatusCarrier>
        <int:CreatedOnDate>2019-09-27T20:20:13.6233824+03:00</int:CreatedOnDate>
        <int:Status>Won</int:Status>
      </int:BetStatusCarrier>
    </int:BetStatus>
    <int:Payout>120</int:Payout>
  </int:UpdateBetSlipRequest.BetCarrier>

  <int:UpdateBetSlipRequest.BetCarrier>
    <int:BetReferenceNumber>2</int:BetReferenceNumber>

    <int:BetStatus>
      <!-- One or more repetitions:-->
      <int:BetStatusCarrier>
        <int:CreatedOnDate>2019-09-27T20:20:13.6233824+03:00</int:CreatedOnDate>
        <int:Status>PaidWon</int:Status>
      </int:BetStatusCarrier>
    </int:BetStatus>

    <int:Payout>100</int:Payout>
  </int:UpdateBetSlipRequest.BetCarrier>

```

```

        </int:Bets>
    </int:UpdateBetSlipRequest>
</tem:betSlips>
</tem:UpdateBetSlips>
</soapenv:Body>
</soapenv:Envelope>
    
```

### 3.4.4.3 Method Response

When the *UpdateBetSlips* method is called, the API responds with the XML structure shown in Figure 3.10. The response provides feedback on each and every bet slip received, by means of its *ReferenceNumber* field. The types of error that might be returned are described in Table 3.29.

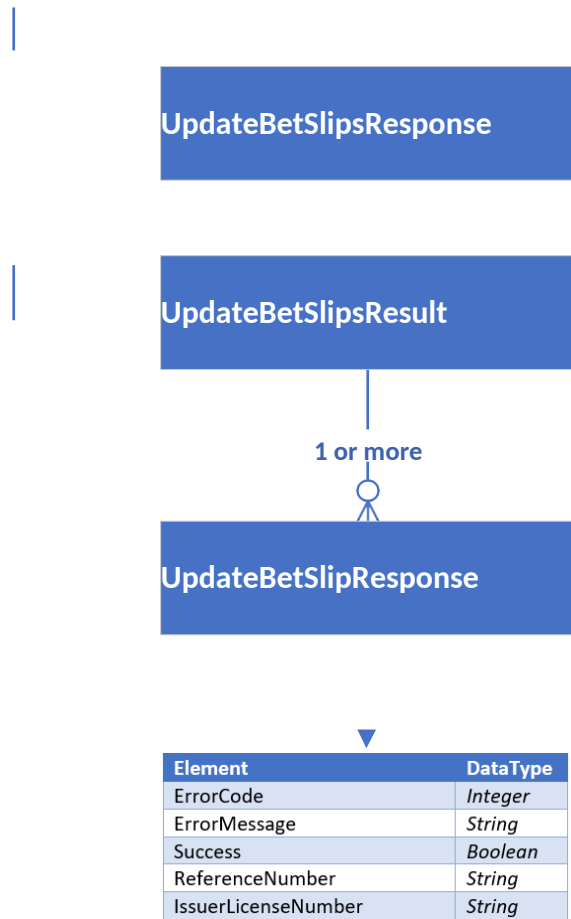


Figure 3.10: The XML structure of *UpdateBetSlipsResponse*.

ErrorCode	ErrorMessage	Description
1500	"No Licensee Found for LicenseNumber:'{0}' LicenseeIdentifier:'{1}' DataEntryKey:'{2}'"	Information about the licensee passed in the header do not match with any licensee in BMRS (E.g. Invalid LicenseNo, DataEntryKey etc.)
1501	"Representative Licensee '{0}' not found"	Unknown representative license
1503	"Licensee {0} does not represents Licensee {1}"	Representative License does not represent bookmaker
1504	"Field '{0}': value is null or empty"	A mandatory value or element is missing (e.g. the header of the request)
1505	"Field '{0}': value '{1}' is invalid"	The value passed is invalid (e.g. an invalid country codes, undefined enumerations, non-expected negative or non-expected zero values)
1506	"Field '{0}': value '{1}' is duplicated"	Duplicate items exist in the same request (e.g. BetSlips, BetSlipItems, BetLines, Accounts, AccountTransactions, Restrictions)
1514	"Licensee {0} does not accept bets. Licensee Type: {1}"	Betslips are allowed to be only for ClassB or SHOP. Class A and representatives must associate their betslips with a shop.

**Table 3.29:** Types of error in UpdateBetSlipsResponse.

The following is an example of the SOAP/XML response:

UpdateBetSlips Sample Response:
<pre> &lt;s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"&gt;   &lt;s:Body&gt;     &lt;UpdateBetSlipsResponse xmlns="http://tempuri.org/"&gt;       &lt;UpdateBetSlipsResult xmlns:a="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Respo nse" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"&gt;         &lt;a:UpdateBetSlipResponse&gt;           &lt;a:ErrorCode&gt;0&lt;/a:ErrorCode&gt;           &lt;a:ErrorMessage i:nil="true"/&gt;           &lt;a:Success&gt;true&lt;/a:Success&gt;           &lt;a:ReferenceNumber&gt;UT-38115dcca4ce4-1277-4c56-b48b-7ac46018123&lt;/a:ReferenceNumber&gt;           &lt;a:RepresentativeLicenseNumber&gt;SHOP-UT-0002&lt;/a:RepresentativeLicenseNumber&gt;         &lt;/a:UpdateBetSlipResponse&gt;       &lt;/UpdateBetSlipsResult&gt;     &lt;/UpdateBetSlipsResponse&gt;   &lt;/s:Body&gt; &lt;/s:Envelope&gt; </pre>

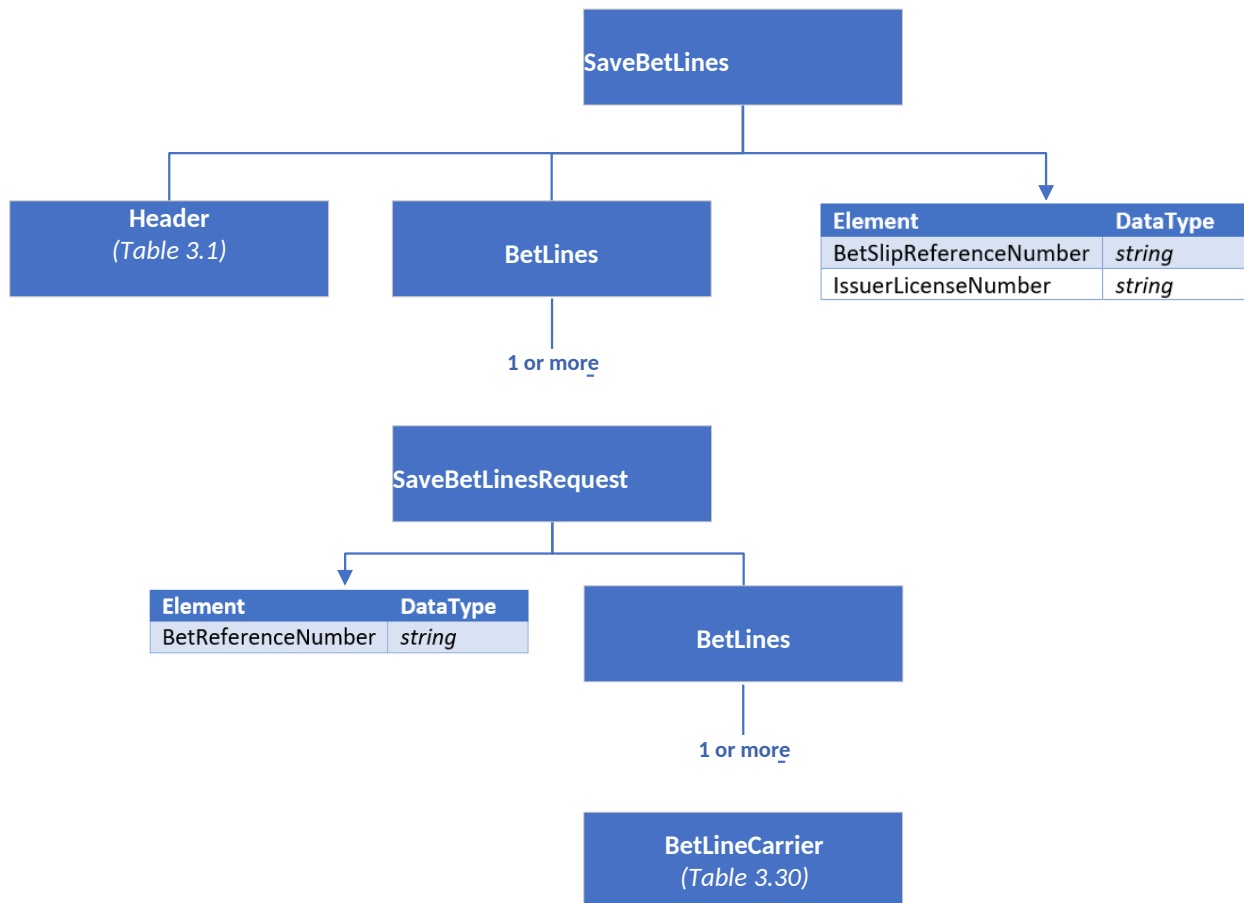
## 3.4.5 Method SaveBetLines

### 3.4.5.1 Description

**Note:** The NBA does not require the Lines of any Bet within the regular 90-second data transmission window. This Method should only be called following a request by the Authority for one or more BetSlips.

The combinations or lines derived from Bets might be required by the NBA under exceptional circumstances where a BetSlip needs to be examined down to its finest detail. To accommodate such cases, the BMRS API provides the *SaveBetLines* function, the structure of which is shown in Figure 3.11. The data fields under the *BetLineCarrier* element are described in Table 3.30.

### 3.4.5.2 Method Request



**Figure 3.11:** The XML structure of the *SaveBetLines* Method.

BetLineCarrier				
No.	Field Name	Data Type	Description	Comment
1	LineReferenceNumber	Integer	A unique identifier among other Lines of this Bet, assigned by the Bookmaker system.	Preferably a sequential integer.
2	BetSlipItemReferenceNumbers	String	A comma-separated string referencing the BetSlipItems of this Line.	
3	Odds	Decimal	The combined odds of this Line.	
4	Stake	Decimal	The amount of stake on this Line. Includes bonus.	

**Table 3.30:** Data fields (sub-elements) under the BetLineCarrier element.

The following is an example of this method's SOAP/XML request:

```

SaveBetLines Sample Request

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/"
xmlns:int="http://schemas.datacontract.org/2004/07/IntelScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Requ
est">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:SaveBetLines>

      <tem:header>
        <int:DataEntryKey>C2EDF452-9B0E-4B75-85BA-D8C95F242371</int:DataEntryKey>
        <int:LicenseNumber>B-UT-0001</int:LicenseNumber>
        <int:LicenseeIdentifier>6254122F-CA27-4034-A629-B855DD0D25EC</int:LicenseeIdentifier>
      </tem:header>

      <tem:betSlipReferenceNumber>UT-38115dcca4ce4-1277-4c56-b48b-7ac460188ff4</tem:betSlipReferenceNumber>
      <tem:IssuerLicenseNumber>B-UT-0001</tem:IssuerLicenseNumber>

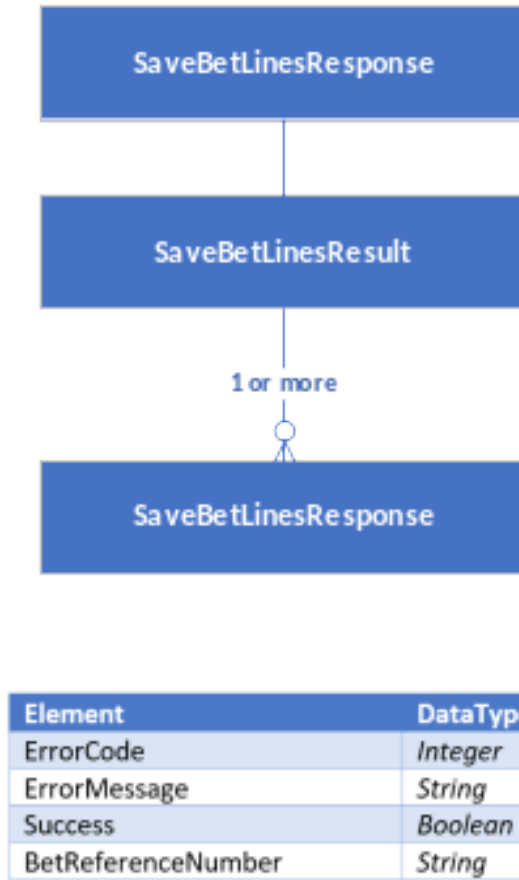
      <tem:betLines>
        <!--One or more repetitions:-->
        <int:SaveBetLinesRequest>
          <int:BetLines>
            <!--One or more repetitions:-->
            <int:SaveBetLinesRequest.BetLineCarrier>
              <int:BetSlipItemReferenceNumbers>1</int:BetSlipItemReferenceNumbers>
              <int:LineReferenceNumber>1</int:LineReferenceNumber>
              <int:Odds>12</int:Odds>
              <int:Stake>100</int:Stake>
            </int:SaveBetLinesRequest.BetLineCarrier>
          </int:BetLines>
          <int:BetReferenceNumber?</int:BetReferenceNumber>
        </int:SaveBetLinesRequest>
      </tem:betLines>
    </tem:SaveBetLines>

  </soapenv:Body>
</soapenv:Envelope>

```

### 3.4.5.3 Method Response

Following a call to the *SaveBetLines* method, the API response structure is shown in Figure 3.12. The types of error that might be returned are described in Table 3.31.



**Figure 3.12:** The XML structure of *SaveBetLinesResponse*.

ErrorCode	ErrorMessage	Description
1500	"No Licensee Found for LicenseNumber:'{0}' LicenseeIdentifier:'{1}' DataEntryKey:'{2}'"	Information about the licensee passed in the header do not match with any licensee in BMRS (E.g. Invalid LicenseNo, DataEntryKey etc.)
1501	"Representative Licensee '{0}' not found"	Unknown representative license
1503	"Licensee {0} does not represents Licensee {1}"	Representative License does not represent bookmaker
1504	"Field '{0}': value is null or empty"	A mandatory value or element is missing (e.g. the header of the request)
1505	"Field '{0}': value '{1}' is invalid"	The value passed is invalid (e.g. an invalid country codes, undefined enumerations, non-expected negative or non-expected zero values)
1506	"Field '{0}': value '{1}' is duplicated"	Duplicate items exist in the same request (e.g. BetSlips, BetSlipItems, BetLines, Accounts, AccountTransactions, Restrictions)
1514	"Licensee {0} does not accept bets. Licensee Type: {1}"	Betslips are allowed to be only for ClassB or SHOP. Class A and representatives must associate their betslips with a shop.

**Table 3.31:** Types of error in SaveBetLinesResponse.

The following is an example of the SOAP/XML response of this method:

SaveBetLines Sample Response:
<pre> &lt;s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"&gt;   &lt;s:Body&gt;     &lt;SaveBetLinesResponse xmlns="http://tempuri.org/"&gt;       &lt;SaveBetLinesResult xmlns:a="http://schemas.datacontract.org/2004/07/InteliScape.NBA.BMRS.BusinessLogic.IntegrationManagement.Carrier.Respo nse" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"&gt;         &lt;a:SaveBetLinesResponse&gt;           &lt;a:ErrorCode&gt;0&lt;/a:ErrorCode&gt;           &lt;a:ErrorMessage i:nil="true"/&gt;           &lt;a:Success&gt;&gt;true&lt;/a:Success&gt;           &lt;a:BetReferenceNumber?&lt;/a:BetReferenceNumber&gt;         &lt;/a:SaveBetLinesResponse&gt;       &lt;/SaveBetLinesResult&gt;     &lt;/SaveBetLinesResponse&gt;   &lt;/s:Body&gt; &lt;/s:Envelope&gt; </pre>

## 3.5 System Calls

This final class of BMRS methods act outside the scope of any betting business activity and are utilized to provide feedback to both ends of the data exchange communication channel, namely the API and the Bookmaker's system.

### 3.5.1 Method Heartbeat

#### 3.5.1.1 Description

A call to the CreateHeartbeat method is a timestamped signal from the Bookmaker's system to the BMRS and confirms that the connection between the two endpoints is alive, which becomes relevant during periods of low traffic, when other methods are no longer called because there is no data to transmit.

**Note:** A periodic call to this function is compulsory, i.e., it is not data-dependent because it does not carry any data other than a timestamp.

#### 3.5.1.2 Method Request

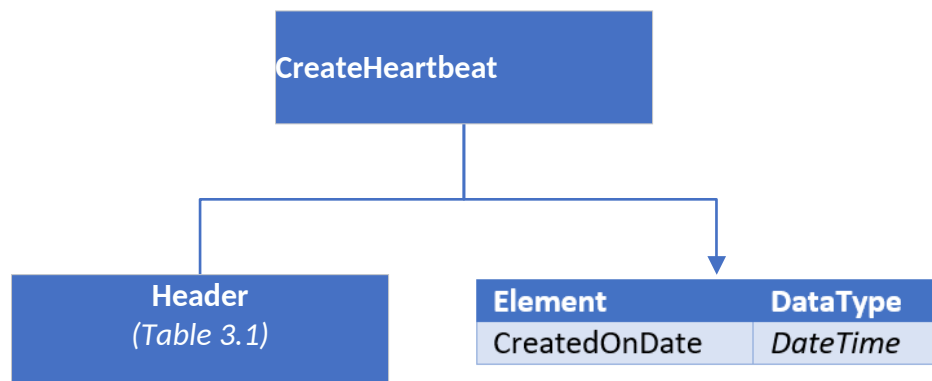


Figure 3.13: The XML structure of the CreateHeartbeat Method.

The following is an example of this method's SOAP/XML request :

HeartBeat Sample Request:
<pre> &lt;soapenv:Envelope xmlns:soapenvtp="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="" xmlns:inttt="http://schemas.xmlsoap.org/soap/envelope/"&gt; &lt;soapenv:Header/&gt; &lt;soapenv:Body&gt; &lt;tem:CreateHeartbeat&gt; &lt;tem:header&gt; &lt;int:DataEntryKey&gt;8F063140-4D72-3F01-1E8C-09C90F86722D&lt;/int:DataEntryKey&gt; &lt;int:LicenseNumber&gt;A-90181&lt;/int:LicenseNumber&gt; &lt;int:LicenseIdentifier&gt;B3411DE4-5E4B-4199-FC08-4C4E0B49C2CF&lt;/int:LicenseIdentifier&gt; &lt;/tem:header&gt; &lt;tem:createdOnDate&gt;2019-09-27T15:35:52.939Z&lt;/tem:createdOnDate&gt; &lt;/tem:CreateHeartbeat&gt; &lt;/soapenv:Body&gt; &lt;/soapenv:Envelope&gt; </pre>

## 3.5.2 Method GetErrorLogs

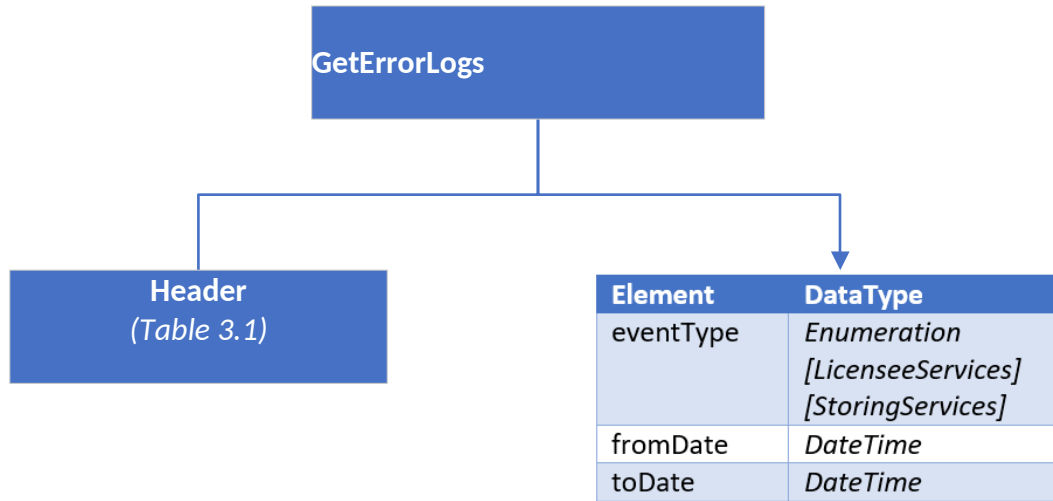
### 3.5.2.1 Description

As previously shown, the BMRS returns a response in every occasion the API is called, and within that response provides feedback for any possible errors that might have been encountered in attempting to process the data package sent. These errors, however, are of limited scope because they are the result of a preliminary data check that takes place during the data queueing process (Licensee Services) and before any operation at the database level is performed (Storing Services). Consequently, the sending system (Bookmaker) is oblivious to any mishaps that might have happened further down the storing process.

The *GetErrorLogs* method aims towards alleviating the limitation described above and provide feedback at a deeper level, on request by the sender. The XML structure for the call to this method is shown in Figure 3.14 and the equivalent response in Figure 3.15. Table 3.34 lists the types of error that might be returned by the latter.

**Note:** Unlike other methods of the API, *GetErrorLogs* does not follow a regular cycle and only gets called when the sender wishes to review any errors.

### 3.5.2.2 Method Request



**Figure 3.14:** The XML structure of the GetErrorLogs Method.

The following is an example of the SOAP/XML request for this method:

```

GetErrorLogs Sample Request:

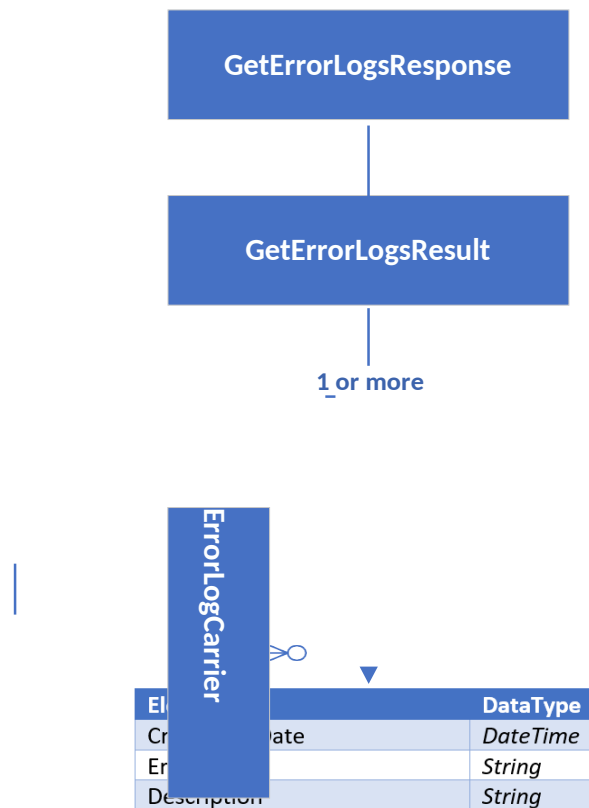
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/" xmlns:inttt="http://tempuri.org/est">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:GetErrorLogs>
      <tem:header>
        <int:DataEntryKey>F2D64583-DFC1-4DF1-A95D-02BCD1CBA489</int:DataEntryKey>
        <int:LicenseNumber>A-UT-0001</int:LicenseNumber>
        <int:LicenseeIdentifier>19666639-C7F8-4942-876D-CA1088B75EFE</int:LicenseeIdentifier>
      </tem:header>
      <tem:eventType>StoringService</tem:eventType>
      <tem:fromDate>2019-09-20T15:00:00.000Z</tem:fromDate>
      <tem:toDate>2019-09-27T15:00:00.000Z</tem:toDate>
    </tem:GetErrorLogs>
  </soapenv:Body>
</soapenv:Envelope>

```

EventType	Description
LicenseeService	Errors detected upon submission to the API.
StoringService	Errors detected during message processing for storing.

**Table 3.33:** Types of event in GetErrorLogs.

### 3.5.2.3 Method Response



**Figure 3.15:** The XML structure of GetErrorLogsResponse.

ErrorCode	ErrorMessage	Description
1500	"No Licensee Found for LicenseNumber:'{0}' LicenseeIdentifier:'{1}' DataEntryKey:'{2}'"	Information about the licensee passed in the header do not match with any licensee in BMRS (E.g. Invalid LicenseNo, DataEntryKey etc.)
1504	"Field '{0}': value is null or empty"	A mandatory value or element is missing (e.g. the header of the request)

**Table 3.34:** Types of LicenseeService error in GetErrorLogsResponse.

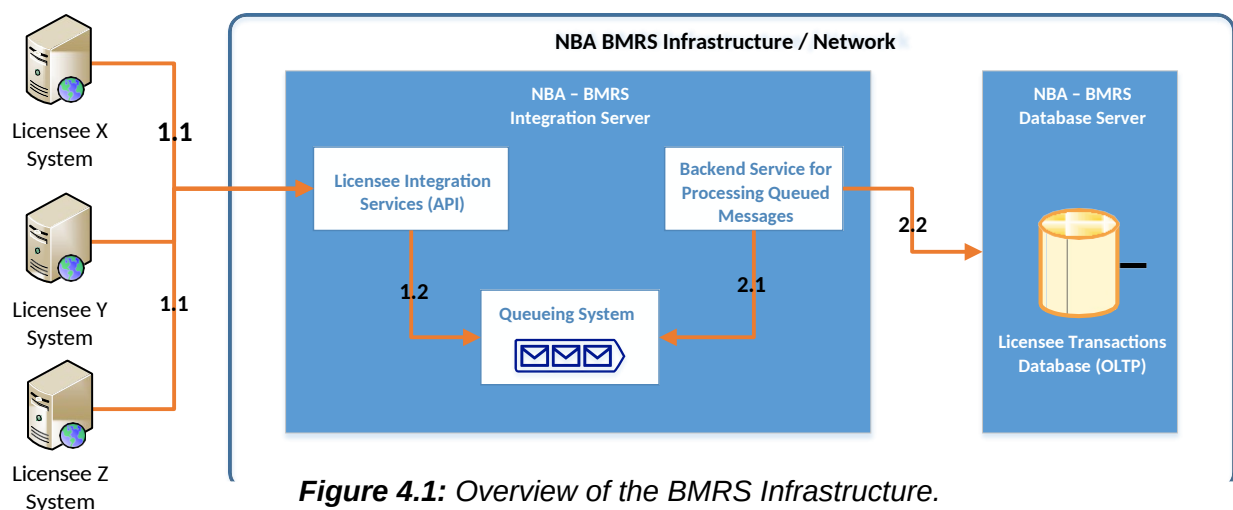
GetErrorLogs Sample Response
<pre> &lt;s:Envelope xmlns:stp"&gt; &lt;s:Body&gt; &lt;GetErrorLogsResponse xmlns:stp"&gt; &lt;GetErrorLogsResult xmlns:a=" nse" xmlns:itt"&gt; &lt;a:ErrorLogCarrier&gt; &lt;a:CreatedOnDate&gt;2019-09-25T15:40:38+03:00&lt;/a:CreatedOnDate&gt; &lt;a:Description&gt;Storing transaction failed. Transaction with Reference Number: 'RefNo1' for Username: 'testxp1' already exist&lt;/a:Description&gt; &lt;a:ErrorCode&gt;1507&lt;/a:ErrorCode&gt; &lt;/a:ErrorLogCarrier&gt; &lt;a:ErrorLogCarrier&gt; &lt;a:CreatedOnDate&gt;2019-09-25T15:40:51+03:00&lt;/a:CreatedOnDate&gt; &lt;a:Description&gt;Storing transaction failed. Transaction with Reference Number: 'RefNo1' for Username: 'testxp1' already exist&lt;/a:Description&gt; &lt;a:ErrorCode&gt;1507&lt;/a:ErrorCode&gt; &lt;/a:ErrorLogCarrier&gt; &lt;/GetErrorLogsResult&gt; &lt;/GetErrorLogsResponse&gt; &lt;/s:Body&gt; &lt;/s:Envelope&gt; </pre>

## 4.0 BMRS Communication Protocols

The communication between a bookmaker's platform and the NBA's BMRS will be supported by a primary and a secondary protocol.

### 4.1 Primary Communication Protocol – The BMRS API

The primary communication protocol takes the form of an application programming interface (API) available over the internet, through which licensed entities push the required data to the NBA within a 90-seconds time window. At the receiving end, a queueing mechanism is utilized by the BMRS to ensure uninterrupted data flow and counter potential bottlenecks in transmission during hours of peak traffic, as illustrated in the diagram below.



**Figure 4.1:** Overview of the BMRS Infrastructure.

The following list enumerates the high-level flow of receiving and storing the data:

- **Step 1.1:** Licensee system sends data to the API
- **Step 1.2:** API validates the data and stores them in the Queueing System. Before exiting, the API returns a response that may contain a list of errors.
- **Step 2.1:** A separate thread runs constantly and reads messages stored in the queue.
- **Step 2.2:** Messages in the queue are processed and stored in the Licensee Transactions Database.

### 4.1.1 Technical Specifications

The API interface protocol employed by the BMRS is the **XML/SOAP 1.1** Web Services, supplemented by communicative metadata over WSDL files to allow the effective discovery and interpretation of the available methods. This protocol is maintained by the “World Wide Web Consortium (W3C)” and relies on application layer protocols that are operating system independent and thus widely adoptable.

#### **About SOAP**

*SOAP is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to induce extensibility, neutrality and independence. It uses XML Information Set for its message format, and relies on application layer protocols, most often Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission. SOAP allows processes running on disparate operating systems (such as Windows and Linux) to communicate using Extensible Mark- up Language (XML). Since web protocols like HTTP are installed and running on all operating systems, SOAP allows clients to invoke web services and receive responses independent of language and platforms.*

#### **About WSDL**

*The Web Services Description Language (WSDL) is an XML-based interface definition language that is used for describing the functionality offered by a web service. The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. Therefore, its purpose is roughly similar to that of a type signature in a programming language.*

It is recommended that developers use the provided WSDL file of BMRS APIs to generate the proxy classes in their systems by using the network location of the WSDL file or by importing the file in their projects. This will enable the correct auto generation of all methods, classes and related fields, thus eliminating possible errors arising from manually creating all xml request and response commands.

### 4.1.2 Authentication/Authorization

Authentication of the systems connected to the BMRS is achieved by means of shared secrets

(an API key – random UUIDs), which needs to be incorporated in the header of every batch request to the web services. It is noted that the web services will be stateless, hence the calls will be authenticated on all requests.

Both authentication and authorization take place at the **license level**, meaning that each company will have a system account per license (e.g. one for class A and one for Class B in case they have both). The licensees will be allowed to send data regarding betting activity that are related to the representative shops associated with them.

#### 4.1.3 Security (TLS)

Transport layer security (TLS), otherwise known as Secure Socket Layer (SSL), is employed to protect the communication channel between the licensee systems and the Web services exposed by the BMRS. TLS will protect the data while they are in transit by encrypting them.

#### 4.1.4 Network Access Rules

The BMRS web services will follow network access rules at the firewall level to limit their exposure to a predefined whitelist of IP addresses. As such, regulated entities will need to report to the Authority the IP addresses of their systems to enable appropriate network access.

## 4.2 Secondary Communication Protocol – XML Files

The secondary integration protocol will take the form of XML Files and should only be employed following consent from the NBA.

A predefined XML schema (XSD) shall be made available to the licensees in order to ensure that the data conforms to a unified standard. Regulated entities will need to prepare the data as per the XSD Schema and send it to NBA IT Officers for import. Regulated entities will need to prepare the data as per the XSD schema at regular intervals, the period of which needs to be agreed with the Authority. These files should then either be hosted in a secure online location to which the NBA is given access for data retrieval following a "pull" protocol or sent to the Authority over alternative communication channels.

The same datasets described in the previous section (Primary Protocol – Web Services) will be

used for the templates of the XML Files.