Betfair API Quick Start and Best Practice Guide

Table of Contents

Document Purpose

This document is aimed at developers who are interested in writing applications that use the Betfair Sports API.

This guide provides details of answers to frequently asked questions and incorporates XML examples of a selection of API calls required to obtain data from Betfair.

A detailed technical description and full details of all Betfair Sports API services can be found in the Betfair Sports Exchange API Reference Guide. It can be downloaded from the Support Center of the BDP Developer Support Site

Web Service Testing Tools

For testing API functions we would recommend using a Web Service testing tool such as SOAP UI (http://www.soapui.org/)

This will allow you to test API functionality using SOAP XML requests.

To set up SOAP UI to use the Betfair API please follow the steps below:

1. Download SOAP UI via http://sourceforge.net/projects/soapui/files/soapui/3.0.1
2. Open up SOAP UI
3. Click on File > New WDSL Project
4. Enter 'Global API' as the 'Project Name' and https://api.betfair.com/global/v3/BFGlobalService.wsdl as 'Initial WDSL'.
5. Click OK
6. Repeat step 3, but enter Exchange API as 'Project Name' and https://api.betfair.com/exchange/v5/BFExchangeService.wsdl as the 'Initial WSDL and click OK.

   For the Australian Exchange please enter https://api-au.betfair.com/exchange/v5/BFExchangeService.wsdl as the 'Initial WSDL and Aus Exchange as the 'Project Name'.

How do I?

The table below shows the most appropriate API service for each task.

If you want to:	Use:
Login	**Login**
Request a list of available events	**getActiveEventTypes**
Request a list of Market for a specific eventType	**getAllMarkets**
Request details of a Market (excluding prices)	**getMarket**
Request prices for a Market	**getMarketPricesCompressed**
Place a bet	**placeBets**
Cancel a bet before it is matched	**cancelBets**
Retrieve a list of my Matched/Unmatched bets	**getMUBets**
Edit an Unmatched bet	**updateBets**
Retrieve the P&L for a market	**getMarketProfitAndLoss**
Place a Betfair SP bet	**placeBets**
Check if a market is in-play now	**getMarketPricesCompressed**
Check if a market is due to be turned in-play	**getAllMarkets**
Retrieve a list of Settled bets	**getBetHistory**
Retrieve your P&L for a market	**getMarketProfitandLoss**

Logging on to the Betfair API

All Betfair API functions require you to login to Betfair to obtain a session token.

The resulting session token **Login** response must then be used in any requests made to subsequent API services.

Login

Parameter	Example Value
username	Username
password	Password
productId	82
vendorsoftwareId	0
locationId	0
Ipaddress	0

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfg="http://www.betfair.com/publicapi/v3/BFGlobalService/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfg:login>
         <bfg:request>
            <ipAddress>0</ipAddress>
            <locationId>0</locationId>
            <password>password</password>
            <productId>82</productId>
            <username>username</username>
            <vendorSoftwareId>0</vendorSoftwareId>
         </bfg:request>
      </bfg:login>
   </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soap:Body>
 <n:loginResponse xmlns:n="http://www.betfair.com/publicapi/v3
BFGlobalService/">
   <n:Result xsi:type="n2:LoginResp">
      <header xsi:type="n2:APIResponseHeader">
        <errorCode xsi:type="n2:APIErrorEnum">OK</errorCode>
        <minorErrorCode xsi:nil="1"/>
        <sessionToken
xsi:type="xsd:string">T0GPCMPcBqmm3uOICzxsapLK76BMZm0uVDLDVWz5U=</sessionToken>
        <timestamp xsi:type="xsd:dateTime">2011-02-07T11:43:47.455Z</timestamp>
     </header>
       <currency xsi:type="xsd:string">GBP</currency>
       <errorCode xsi:type="n2:LoginErrorEnum">OK</errorCode>
       <minorErrorCode xsi:nil="1"/>
       <validUntil xsi:type="xsd:dateTime">2011-02-
10T00:00:00.000Z</validUntil>
   </n:Result>
 </n:loginResponse>
</soap:Body>
```

Requesting a list of available Events

Taking the **sessionToken** from the Login response, you can make a request using the **getActiveEventTypes** service which will return a response containing the **eventTypes** (e.g. Soccer, Horse Racing etc) that are currently available on Betfair.

getActiveEventTypes

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
Locale	en

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfg="http://www.betfair.com/publicapi/v3/BFGlobalService/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfg:getAllEventTypes>
         <bfg:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken> //From Login
response//
            </header>
            <locale>en</locale> //English Language//
         </bfg:request>
      </bfg:getAllEventTypes>
   </soapenv:Body>
</soapenv:Envelope>
```

Requesting a list of Markets for a specific eventType

After determining the **eventTypeId/s** that you are interested in from the response of the **getActiveEventTypes** call, you can use the **getAllMarkets** API call to obtain a list of markets that belong to that **eventTypeId.**

For example, for a list of Soccer markets which are based in Great Britain, you would use the parameters shown below. **Please note:** the **getAllMarkets** response is cached and updated every 5 minutes.

getAllMarkets

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
locale	en
eventTypeId	1
countries	GBR
fromDate	2010-12-10T00:00:00.000Z
toDate	2010-12-30T00:00:00.000Z

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/"
xmlns:v5="http://www.betfair.com/publicapi/types/exchange/v5/">
  <soapenv:Header/>
    <soapenv:Body>
      <bfex:getAllMarkets>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>  //From Login
response//
            </header>
            <locale>en</locale>  //English language//
            <eventTypeIds>
               <v5:int>1</v5:int>  //Soccer//
            </eventTypeIds>
            <countries>
               <v5:Country>GBR</v5:Country>
            </countries>
            <fromDate>2010-12-10T00:00:00.000Z</fromDate>  //Change to valid
timeframe//
            <toDate>2010-12-30T00:00:00.000Z</toDate>      //Change to valid
timeframe//
         </bfex:request>
      </bfex:getAllMarkets>
   </soapenv:Body>
</soapenv:Envelope>
```

Requesting details of a specific Market (excluding prices)

Using the **marketId** returned in the **getAllMarkets** API response, you can obtain further information about the specific market, including the **selectionId** for each outcome using the **getMarket** service.

getMarket

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
locale	en
marketId	12345678
includeCouponLinks	FALSE

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:getMarket>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <locale>en</locale>
            <marketId>1234567</marketId>
            <includeCouponLinks>FALSE</includeCouponLinks>
         </bfex:request>
      </bfex:getMarket>
   </soapenv:Body>
</soapenv:Envelope>
```

Requesting prices for a Market

Once you have identified the market (marketId) that you are interested using the **getAllMarkets** & **getMarket** service, you can request prices for that market using the **getMarketPricesCompressed** API call.

getMarketPricesCompressed

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
currencyCode	GBP
marketId	12345678

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:getMarketPricesCompressed>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <currencyCode>GBP</currencyCode>
            <marketId>12345678</marketId>
         </bfex:request>
      </bfex:getMarketPricesCompressed>
   </soapenv:Body>
</soapenv:Envelope>
```

Placing a Bet

To place a bet you require the **marketId** and **selectionId** from the **getMarket** API call. The below parameters will place a normal Exchange bet at odds of 1000 for a stake of £2.0.

If the bet is placed successfully, a betId is returned in the PlaceBetsResult respone. If the response contains sizeMatched equal to zero, the bet has not been immediately matched. You can use the getMUBets service to check the status of your bet.

placeBets

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
asianlineId	0
betType	B
betCategoryType	E
betPersistenceType	NONE
marketId	12345678
price	1000
selectionId	87654321
size	2.0
bspLiability	0

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/"
xmlns:v5="http://www.betfair.com/publicapi/types/exchange/v5/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:placeBets>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <bets>
               <v5:PlaceBets>
                  <asianLineId>0</asianLineId>
                  <betType>B</betType>
                  <betCategoryType>E</betCategoryType>
                  <betPersistenceType>NONE</betPersistenceType>
                  <marketId>12345678</marketId>
                  <price>1000</price>
                  <selectionId>47972</selectionId>
                  <size>2.0</size>
                  <bspLiability>0</bspLiability>
               </v5:PlaceBets>
            </bets>
         </bfex:request>
      </bfex:placeBets>
   </soapenv:Body>
</soapenv:Envelope>
```

Cancelling a Bet

To cancel an unmatched bet, or a number of bets in one market, use the cancelBets API service. You can submit one or more betIds from a single market in the request.

Note: You cannot cancel a bet once it has been matched.

cancelBet

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
betId	12345678

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/"
xmlns:v5="http://www.betfair.com/publicapi/types/exchange/v5/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:cancelBets>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <bets>
               <!--Zero or more repetitions:-->
               <v5:CancelBets>
                  <betId>13820312693</betId>
               </v5:CancelBets>
            </bets>
         </bfex:request>
      </bfex:cancelBets>
   </soapenv:Body>
</soapenv:Envelope>
```

Retrieve a List of Matched/Unmatched bets

To check the status of the bet/s you have placed you should use the **getMUBets** API call and use the **betStatus** 'MU'. You should specify the **marketId** in the **getMUBets** request when possible. **getMUBet** requests without a **marketId** specified in the requests are counted as 5 data requests for the purpose of calculating the Data Request Charge

getMUBets

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
betStatus	MU
marketId	12345678
betid	0
orderBy	MATCHED_DATE
sortOrder	ASC
startRecord	0
matchedSince	2010-12-10T08:22:00.120Z
excludeLastSecond	true

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/"
xmlns:v5="http://www.betfair.com/publicapi/types/exchange/v5/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:getMUBets>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <betStatus>MU</betStatus>   //'MU' returns both 'Matched' and
'Unmatched' bets//
           <marketId>102088158</marketId>
            <orderBy>MATCHED_DATE</orderBy>
            <sortOrder>ASC</sortOrder>
            <recordCount>100</recordCount> //Maximum record count is 100//
              <startRecord>0</startRecord>
            <matchedSince>2010-12-10T08:22:00.120Z</matchedSince>  //Change to
valid timeframe//
            <excludeLastSecond>true</excludeLastSecond>
         </bfex:request>
      </bfex:getMUBets>
   </soapenv:Body>
</soapenv:Envelope>
```

Editing an 'Unmatched Bet'

To edit a bet, you require the betId of the original bet from the **placeBets** or **getMUBets** response.

Please note: **You can only update the bet size or the bet Price in a single request.**

The below example will result in a bet being updated from a price of 4.0 to 5.0 for a stake of £2.0

updateBets

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
betId	12345678
newPrice	5.0
newSize	2.0
oldPrice	4.0
oldSize	2.0
oldBetPersistenceType	NONE
newBetPersistanceType	NONE

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/"
xmlns:v5="http://www.betfair.com/publicapi/types/exchange/v5/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:updateBets>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <bets>
               <!--Zero or more repetitions:-->
               <v5:UpdateBets>
                  <betId>13165057028</betId>
                  <newPrice>5.0</newPrice>    // Only price or size of bet can
be changed.  You cannot change both price and size in one request//
                  <newSize>2.0</newSize>
                  <oldPrice>4.0</oldPrice>
                  <oldSize>2.0</oldSize>
                  <oldBetPersistenceType>NONE</oldBetPersistenceType>
                  <newBetPersistenceType>NONE</newBetPersistenceType>
               </v5:UpdateBets>
            </bets>
         </bfex:request>
      </bfex:updateBets>
   </soapenv:Body>
</soapenv:Envelope>
```

Placing a Betfair SP bet via the Betfair API

To place a bet on a selection at Betfair SP, you need to specify the parameters below in the **placeBets** request. The below example would place a Betfair SP back bet on the required selection for a stake of £2.00.

PlaceBet – Betfair SP

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
asianLineId	0
betType	B
betCategoryType	E
betPersistenceType	SP
marketid	12345678
price	2.0
selectionId	12345
size	2.0
bspLiability	0

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/"
xmlns:v5="http://www.betfair.com/publicapi/types/exchange/v5/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:placeBets>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <bets>
               <v5:PlaceBets>
                  <asianLineId>0</asianLineId>
                  <betType>B</betType>              //Back bet//
                  <betCategoryType>E</betCategoryType>
                  <betPersistenceType>SP</betPersistenceType>  //SP bet//
                  <marketId>102090122</marketId>
                  <price>2</price>
                  <selectionId>1528775</selectionId>
                  <size>2.0</size>
                  <bspLiability>0</bspLiability>
               </v5:PlaceBets>
            </bets>
         </bfex:request>
      </bfex:placeBets>
   </soapenv:Body>
</soapenv:Envelope>
```

Retrieving your Profit/Loss for a Market

The **getMarketProfitandLoss** API service allows you to retrieve your profit and loss for each selection within a Betfair market.

You have the option whether to include bets that have already been settled or to include the figures with commission already applied.

The below request will exclude 'Settled' bets and commission from the calculation.

getMarketProfitAndLoss

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
Locale	en
includeSettledBets	FALSE
includeBSPBets	FALSE
marketId	En
netOfCommission	FALSE

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:getMarketProfitAndLoss>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <locale>en</locale>
            <includeSettledBets>TRUE</includeSettledBets>
            <includeBspBets>FALSE</includeBspBets>
            <marketID>102226096</marketID>
            <netOfCommission>FALSE</netOfCommission>  //Profit will be shown
before commission//
         </bfex:request>
      </bfex:getMarketProfitAndLoss>
   </soapenv:Body>
</soapenv:Envelope>
```

Retrieving a list of Settled Bets

The example demonstrates an API call to the **getBetHistory** service for '**Settled**' bets for Horse Racing specifying the maximum record count of 100.

You can specify more than one **eventTypeId** in a request if you are looking to obtain your settled bets across a number of sports. The below request is for 'Settled' horse racing bets only.

getBetHistory

Parameter	Example Value
clientStamp	0
sessionToken	nv9RZY8f08fPdy7Zmu4fEkmWjTzWVUgBbGRz0AtFE=
betTypesIncluded	S
detailed	TRUE
eventTypeIds	7
locale	en
timezone	GMT
marketTypesIncluded	O
placeDateFrom	2011-02-01T00:00:00.000Z
placeDateTo	2011-02-10T00:00:00.000Z
recordCount	100
sortBetsBy	NONE
startRecord	0

XML Sample

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:bfex="http://www.betfair.com/publicapi/v5/BFExchangeService/"
xmlns:v5="http://www.betfair.com/publicapi/types/exchange/v5/">
   <soapenv:Header/>
   <soapenv:Body>
      <bfex:getBetHistory>
         <bfex:request>
            <header>
               <clientStamp>0</clientStamp>
               <sessionToken>sessionToken</sessionToken>
            </header>
            <betTypesIncluded>S</betTypesIncluded>
 //Settled Bets//
            <detailed>TRUE</detailed>
            <eventTypeIds>
               <v5:int>7</v5:int>
 //Horse Racing//
            </eventTypeIds>
            <locale>en</locale>
            <timezone>GMT</timezone>
            <marketTypesIncluded>
               <v5:MarketTypeEnum>O</v5:MarketTypeEnum>  //Odds market//
            </marketTypesIncluded>
            <placedDateFrom>2011-02-01T00:00:00.000Z</placedDateFrom>
            <placedDateTo>2011-02-10T00:00:00.000Z</placedDateTo>
            <recordCount>100</recordCount>           //Max is 100//
            <sortBetsBy>NONE</sortBetsBy>
            <startRecord>0</startRecord>
         </bfex:request>
      </bfex:getBetHistory>
   </soapenv:Body>
</soapenv:Envelope>
```

Best Practice FAQs

What is the Betfair Market Life Cycle?

A Betfair Market life cycle contains the following states

INACTIVE > ACTIVE <> SUSPENDED > CLOSED

Once 'Active' a Betfair market can be suspended at any time, typically due to a pending action such as a goal scored during an in-play match or due to the removal of a non runner from a horse race.

You should refer to the marketDescription field within the output of the getMarket call for the rules affecting any Betfair Market.

I have VB.NET/C# how do I add the BetfairAPI?

To create Windows application:

1. On the File menu, point to New, and then click Project to open the New Project dialog box.
2. Expand the Visual Basic Projects folder.
3. Click the Windows Application icon.
4. Click OK to create the project.
5. From the Toolbox, drag a Label and a Button to the design surface of Form1 and arrange them to your liking.
6. On the Project menu, click Add Web Reference.
7. In the URL box of the Add Web Reference dialog box, type the URL https://api.betfair.com/global/v3/BFGlobalService.wsdl
8. Click the Go button to retrieve information about the XML Web service.
9. In the Web reference name box, rename the Web reference appropriately.
10. Click Add Reference to add a Web reference for the target XML Web service.

You will need to follow the above process for the two exchange WSDL's:

https://api.betfair.com/exchange/v5/BFExchangeService.wsdl
https://apiau.betfair.com/exchange/v5/BFExchangeService.wsdl

How does the Betfair API cache data?

The two relevant types of caching we use are:

•caches that are updated by an external mechanism (external to that cache), a.k.a delta caching

•caches that are updated by some internal mechanism (internal to that cache)

Delta Caches

The API servers cache pricing data using a delta cache. Whenever any event occurs on the exchange (matching a bet, cancelling a bet, etc.) that changes pricing data, the exchange pushes an update to the cache. This cache is, therefore, always as up to date as possible. Updates to all API servers happen at the same time, so no API server is ever out of date regarding pricing data.

The exchange itself matches and processes bets on a ~200ms cycle and any changes are pushed to the delta caches. This means, of course, that if you request pricing data less than 200ms after a previous change, you cannot possibly get a different response as the exchange would not have completed a bet processing cycle.

All API calls that return pricing data are using the delta cache and are always as live as possible.

Other Caches

Change on the exchange that are not as common are cached using a polling mechanism. For example, when a runner is removed there are quite a few things that need to happen in the system as all unmatched bets on the runner need to be identified and cancelled, all matched bets need to be identified and voided, the funds need to be made available to user's account etc. before the runner can be physically removed from the market. Suspending a market follows a similar pattern of events.

Ideally we would like to provide all information via a delta cache, but it may be some time before this can be achieved.

What is the algorithm for displaying prices inline with the website?

For the available to back prices on the LHS:

- For each runner calculate the best price at which you could effectively back that runner by laying all the others:

1 / (N –∑ 1/(best to lay price of other runners))
Where N is the number of winners that market will have, e.g. for a "to win" market N = 1, for a place market with 3 winners N = 3.

- For each runner calculate the available stake at that price as:

MIN(odds1*stake1, odds2*stake2…oddsn*staken) / [the price calculated in step a. above]
In plain English the stake you can match is limited by the payout (odds*stake) available on each of the component bets that you'd need to match.

- Repeat from the first step above

One of the easiest ways to understand how we generate virtual bets for cross matching is to work through a couple of examples.

Available to Back

Consider the following market and what would happen if we placed a very large back bet at 1.01 on The Draw:

Selections (3)	126.7%		Back	Lay		93.3%
Newcastle	1.01 £999	1.5 £200	1.5 £300	2 £120	2.5 £75	1000 £2
Chelsea	1.01 £999	2.4 £250	2.5 £150	3 £150	20 £10	1000 £2
The Draw	1.01 £999	3 £250	5 £150	10 £100	50 £50	1000 £2

Without cross matching, this bet would be matched in three portions; 1) £150 at 5.0, 2) £250 at 3.0, and £999 at 1.01 with anything remaining being left unmatched. With cross matching we can do better. We have a back bet on Newcastle for £120 at 2.0 and a back bet on Chelsea for £150 at 3.0 (shown in pink on the available to lay side of the market). These two bets can be matched against a back bet on The Draw at a price of 6.0, since 2.0, 3.0, and 6.0 form a 100% book. To ensure that the book is balanced, we choose the stakes to be inversely proportional to the prices. This means that we take the full £120 at 2.0 on Newcastle, only £80 at 3.0 on Chelsea, and **£40 at 6.0 on The Draw, which is the first virtual bet**. Assuming these bets got matched, the market would look like this:

Selections (3)	126.7%		Back	Lay		83.3%
Newcastle	**1.01**	**1.5**	**1.5**	**2.5**	**1000**	
	£999	£200	£300	£75	£2	
Chelsea	**1.01**	**2.4**	**2.5**	**3**	**20**	**1000**
	£999	£250	£150	£70	£10	£2
The Draw	**1.01**	**3**	**5**	**10**	**50**	**1000**
	£999	£250	£150	£100	£50	£2

We now have a back bet on Newcastle for £75 at 2.5 and a back bet on Chelsea for £70 at 3.0 (again shown in pink on the available to lay side of the market). These two bets can be matched against a back bet on The Draw at a price of 3.75, since 2.5, 3.0, and 3.75 also form a 100% book. Balancing the stakes means that we need to take the full £75 at 2.5 on Newcastle, only £62.50 at 3.0 on Chelsea, and **£50 at 3.75 on The Draw, which is the second virtual bet**.

Since 3.75 is less than 5.0, the £150 at 5.0 would be matched first followed by £50 at 3.75. If we continued this process we would get further matching at 1.50 and 1.05, but for the purposes of displaying the market view we have the best 3 prices for the available to back bets on The Draw, and so we can stop calculating the virtual bets. The virtual bets are just the bets that would have been matched had we received a sufficiently large back bet at 1.01; in this example, £40 at 6.0 and £50 at 3.75. We take these virtual bets and merge them with the existing bets on the market to generate the following market view (with the virtual bets shown in green):

Selections (3)	126.7%		Back	Lay		93.3%
Newcastle	**1.01**	**1.5**	**1.5**	**2**	**2.5**	**1000**
	£999	£200	£300	£120	£75	£2
Chelsea	**1.01**	**2.4**	**2.5**	**3**	**20**	**1000**
	£999	£250	£150	£150	£10	£2
The Draw	**3.75**	**5**	**6**	**10**	**50**	**1000**
	£50	£150	£40	£100	£50	£2

The process is repeated to obtain the virtual lay bets (available to back bets) for Newcastle and Chelsea.

Available to Lay

Here we have a slightly different market (as before, chosen to make the numbers nice) and consider what would happen if we placed a very large lay bet at 1000 on The Draw:

Selections (3)	103.3%		Back	Lay		55.0%
Newcastle	**1.01** £999	**1.5** £200	**2** £300	**4** £120	**15** £75	**1000** £2
Chelsea	**1.01** £999	**2.4** £250	**3** £150	**5** £150	**20** £10	**1000** £2
The Draw	**1.01** £999	**3** £250	**5** £150	**10** £100	**50** £50	**1000** £2

Without cross matching, this bet would be matched in three portions; 1) £100 at 10.0, 2) £50 at 50.0, and £2 at 1000 with anything remaining being left unmatched. With cross matching we can do better. We have a lay bet on Newcastle for £300 at 2.0 and a lay bet on Chelsea for £150 at 3.0 (shown in blue on the available to back side of the market). These two bets can be matched against a lay bet on The Draw at a price of 6.0, since 2.0, 3.0, and 6.0 form a 100% book. To ensure that the book is balanced, we choose the stakes to be inversely proportional to the prices. This means that we take only £225 at 2.0 on Newcastle, the full £150 at 3.0 on Chelsea, and **£75 at 6.0 on The Draw, which is the first virtual bet**. Assuming these bets got matched, the market would look like this:

Selections (3)	111.7%		Back	Lay		55.0%
Newcastle	**1.01** £999	**1.5** £200	**2** £75	**4** £120	**15** £75	**1000** £2
Chelsea		**1.01** £999	**2.4** £250	**5** £150	**20** £10	**1000** £2
The Draw	**1.01** £999	**3** £250	**5** £150	**10** £100	**50** £50	**1000** £2

We now have a lay bet on Newcastle for £75 at 2.0 and a lay bet on Chelsea for £250 at 2.4 (again shown in blue on the available to back side of the market). These two bets can be matched against a lay bet on The Draw at a price of 12.0, since 2.0, 2.4, and 12.0 also form a 100% book. Balancing the stakes means that we need to take the full £75 at 2.0 on Newcastle, only £62.50 at 2.4 on Chelsea, and **£12.50 at 12.0 on The Draw, which is the second virtual bet**. This leaves the following market:

Selections (3)	128.3%		Back	Lay		55.0%
Newcastle		**1.01** £999	**1.5** £200	**4** £120	**15** £75	**1000** £2
Chelsea		**1.01** £999	**2.4** £187.50	**5** £150	**20** £10	**1000** £2
The Draw	**1.01** £999	**3** £250	**5** £150	**10** £100	**50** £50	**1000** £2

This time we can't continue the process since there is no valid price for a virtual bet on The Draw that would result in a 100% book, and so we can stop calculating the virtual bets. Again, the virtual bets are just the bets that would have been matched had we received a sufficiently large lay bet at 1000;

in this example, £75 at 6.0 and £12.50 at 12.0.

We take these virtual bets and merge them with the existing bets on the market to generate the following market view (with the virtual bets shown in orange):

Selections (3)	103.3%		Back	Lay		61.7%
Newcastle	**1.01** £999	**1.5** £200	**2** £300	**4** £120	**15** £75	**1000** £2
Chelsea	**1.01** £999	**2.4** £250	**3** £150	**5** £150	**20** £10	**1000** £2
The Draw	**1.01** £999	**3** £250	**5** £150	**6** £75	**10** £100	**12** £12.50

How do I determine if a market has turned in-play?

The best way to identify when the market has been turned in play is to use the 'delay' field that is returned by the getMarketPricesCompressed call. If the returned value does not equal 0, this indicates that all bets placed are subject to a delay and therefore that the market is in-play. For the match you have mentioned this value currently equals 5, indicating a 5 second delay when placing bets.

How do I programmatically find new market IDs of a specific type (i.e. Match Odds)?

The getAllMarkets call is the best way. For example, suppose you wanted to find new American Football markets.

Start by passing in the eventTypeId for (American Football (6423). The API returns all markets of that type in a compressed string format. As follows:

21322866~Match Odds~O~ACTIVE~1226538000000~\American Football\NCAAF 2008/09\Fixtures 12 November\Kent St v Temple~/6423/10148663/26388977/26388978/21322866~0~1~USA~1226397827243~2~1~0.0~N~Y

21322869~Match Odds~O~ACTIVE~1226538000000~\American Football\NCAAF 2008/09\Fixtures 12 November\N Illinois v Central Michigan~/6423/10148663/26388977/26388979/21322869~0~1~USA~1226397827243~2~1~0.0~N~Y

21324517~Winning Margin~O~ACTIVE~1226858400000~\American Football\NFL Season 08/09\Fixtures 16 November\Chicago @ Green Bay~/6423/16561465/26389207/26389222/21324517~0~1~USA~1226397827243~7~1~0.0~N~Y

21322873~Match Odds~O~ACTIVE~1226620800000~\American Football\NCAAF 2008/09\Fixtures 13 November\Akron v Buffalo~/6423/10148663/26388980/26388981/21322873~0~1~USA~1226397827243~2~1~0.0~N~Y

Each market has been separated above (the response contains each market separated by a colon (:). For each market, the fields are separated by a tilde' (~). The first field is the market ID, the third is the market type (for Match Odds markets, this will be 'O'). The 6th field is the textual menu path from the root of the tree. So, you should be able to parse that and find all the match odds markets that correspond to the NFL Season 08/09.

How do I count requests for the purpose of Data Request charging?

The Data Request Charge (DRC) is a charge intended to curb excessive calls to the Betfair Sports API. This article provides details about how exactly requests are counted. More information about the exact charges and rules can be found here.

You can make 20 requests every second without being charged. Betfair counts requests by logging the time and date that a request is received. Then, the number of requests received in any one second are added together. It is important to note that seconds are counted as wall-clock seconds, i.e. the time between one second and the next on a clock, not as a period of time lasting 1000ms. In other words, if you make 30 requests between 12:02:00 and 12:02:01, you will have exceeded the limit by 10 requests.

If you are attempting to calculate the acceptable rate, you should count requests from the start of a real clock second, rather than from an arbitrary point lasting 1 second.

Please also keep in mind that the nature of the Internet means that we suggest you limit usage to 15 requests per second as the default. If you are betting very little or not at all this can help you to avoid a charge.

Finally, if you want to synchronise your request counting with Betfair's time, please use a publicly available Network Time Protocol server to synchronise your computer’s clock to a known reference time.

What is HTTP Compression?

HTTP compression is a capability built into both web servers and web clients to reduce the number of bytes transmitted in an HTTP response. This makes better use of available bandwidth and increases performance while reducing download time. When enabled, HTTP protocol data is compressed before it is sent from the server. Clients capable of receiving compressed HTTP data announce that they support compression in the HTTP header. Almost all modern web browsers support HTTP Compression by default.

The Betfair SOAP API uses HTTP to handle communication between API clients and servers. Therefore, the SOAP messages can be compressed using the same HTTP compression used by web browsers. Custom API applications may need some modification before they can take advantage of this feature. Specifically, they need to send an additional HTTP header to indicate they support receipt of compressed responses from the API. In addition, some environments require you to explicitly decompress the response.

Example GZIP SOAP Request

When an API client sends a SOAP request, it opens a connection to the API server and sends an HTTP request containing the SOAP XML document. A typical HTTP request header looks like this:

POST /global/v3/BFGlobalService HTTP/1.1

User-Agent: Mozilla/4.0

Content-Type: text/xml; charset=utf-8

SOAPAction: "login"

Content-Length: 1011

Host: api.betfair.com

Accept-Encoding: gzip

With this request, the API client asks for the SOAP action "login" on host "api.betfair.com". The client identifies itself as "Mozilla/4.0" and claims that it can receive HTTP responses in gz ip format.

After parsing and processing the client's request, the API server may send the HTTP response in compressed format. A typical HTTP response header looks like this:

HTTP/1.1 200 OK

Server: Apache

Content-Type: text/xml; charset=UTF-8

Content-Encoding: gzip

Content-Length: 2223

[GZIP COMPRESSED DATA]

The response header "Content-Encoding" informs the browser that the following data is compressed with gzip.

Enabling Gzip Compression

C Sharp (C#)

Gzip compression can be enabled by switching on the EnableDecompression property of your web service proxy object. The example enables compressed responses from the Betfair Global server:

```
BFGlobalService service = new BFGlobalService();

service.EnableDecompression = true;
```

Java and Axis 1.2

Gzip compression can be enabled by setting Axis client Call object's HTTPConstants MC_ACCEPT_GZIP and MC_GZIP_RESPONSE to true. The following example is from the Betfair Java sample application:

```
private static BFGlobalServiceStub stub;

// Lazy load the Global service stub generated by Apache Axis.

// This stub is used to make all requests to the Betfair Global API

// The global API is generally used for account management features

private static BFGlobalServiceStub getStub() throws Exception {

    if (stub == null)

    {

        stub = new BFGlobalServiceStub("https://api.betfair.com/global/v3/BFGlobalService");

        // You may set up the connection parameters of the stub here if necessary

        // For example: Wait 20 seconds for a response from the API

        stub._getServiceClient().getOptions().setTimeOutInMilliSeconds(20 * 1000);


stub._getServiceClient().getOptions().setProperty(org.apache.axis2.transport.http.HTTPConstants.MC
_ACCEPT_GZIP, "true");


stub._getServiceClient().getOptions().setProperty(org.apache.axis2.transport.http.HTTPConstants.MC
_GZIP_RESPONSE, "true");

    }

    return stub;

}
```

Perl

Using Perl and the LWP package, you enable gzip compression by setting the gzip request header on the request object:

```
my $ua = LWP::UserAgent->new();

my $request = HTTP::Request->new(POST => $serviceurl);

$request->header(SOAPAction => "$serviceurl");

$request->header(Accept_Encoding => "gzip");
```

The response must be decoded as in the following example

```
...

my $resp = $ua->request($request);

my $content=$resp->decoded_content;
```

Useful Resources and Articles

Betfair Developers Forum (http://forum.bdp.betfair.com/)

Betfair API Reference Guide (http://bdphelp.betfair.com/API6/6.0/RefGuide/wwhelp/wwhimpl/js/html/wwhelp.htm)

Betfair API Sample Code (http://bdp.betfair.com/index.php?option=com_weblinks&catid=59&Itemid=116)

Technical Articles

Turbocharging .Net Webservice Clients - Author Russell Gray

C# 3.0, Parallel LINQ, And The Betfair API - An Introduction - Author Russell Gray