**[Non-Interactive (bot) login](https://api.developer.betfair.com/services/webapps/docs/display/1smk3cen4v3lu3yomq5qye0ni/Non-Interactive+%28bot%29+login)**

[Skip to end of metadata](https://api.developer.betfair.com/services/webapps/docs/display/1smk3cen4v3lu3yomq5qye0ni/Non-Interactive+%28bot%29+login#page-metadata-end)

* Added by [Gordon Mack](https://api.developer.betfair.com/services/webapps/docs/display/%7Emackg), last edited by [Jamie Ingilby](https://api.developer.betfair.com/services/webapps/docs/display/%7Eingilbyj) on Nov 03, 2014  ([view change](https://api.developer.betfair.com/services/webapps/docs/pages/diffpages.action?pageId=3834919&originalId=4392641))

[Go to start of metadata](https://api.developer.betfair.com/services/webapps/docs/display/1smk3cen4v3lu3yomq5qye0ni/Non-Interactive+%28bot%29+login#page-metadata-start)

The non-interactive login method for API-NG requires that you create and upload a self-signed certificate which will be used, alongside your username and password to authenticate your credentials and generate a session token.

For the purposes of this guide, we have used openssl to generate this client, details of which can be found at[**http://www.openssl.org/**](http://www.openssl.org/)

**Getting Started**

There are a couple of steps required before we can actually log in:

1. Create a self-signed certificate
2. Link the certificate to your Betfair account

**Creating a Self Signed Certificate**

API-NG requires that a 1024-bit or 2048-bit RSA certificate be used. There are various tutorials available on the Internet but be aware that the certificate needs to be for client authentication (most tutorials only cover server authentication).

**Create a public/private RSA key pair using openssl**

|  |
| --- |
| openssl genrsa -out client-2048.key 2048 |

**Update or Create the openssl configuration file (openssl.cnf) for OpenSSL to override some of the default settings:**

|  |  |
| --- | --- |
| [ ssl\_client ]  basicConstraints = CA:FALSE  nsCertType = client  keyUsage = digitalSignature, keyEncipherment  extendedKeyUsage = clientAuth | |
| https://api.developer.betfair.com/services/webapps/docs/images/icons/emoticons/warning.png | In Windows, the config file is located in the installation directory of OpenSSL  In Linux distributions, the config file is located at /usr/lib/ssl/openssl.cnf or /etc/ssl/openssl.cnf |

**Create a certificate signing request (CSR).**

|  |
| --- |
| openssl req -new -config openssl.cnf -key client-2048.key -out client-2048.csr      Country Name (2 letter code) [AU]:GB  State or Province Name (full name) [Some-State]:London  Locality Name (eg, city) []:London  Organization Name (eg, company) [Internet Widgits Pty Ltd]:yourcompany.com  Organizational Unit Name (eg, section) []:Security Team  Common Name (e.g. server FQDN or YOUR name) []:Test API-NG Certificate  Email Address []:my.name@mydomain.com    Please enter the following 'extra' attributes  to be sent with your certificate request  A challenge password []:  An optional company name []: |

**Self-sign the certificate request to create a certificate**

|  |  |
| --- | --- |
| openssl x509 -req -days 365 -in client-2048.csr -signkey client-2048.key -out client-2048.crt -extfile openssl.cnf -extensions ssl\_client | |
| https://api.developer.betfair.com/services/webapps/docs/images/icons/emoticons/warning.png | In Windows, using any text editor, copy the contents of the .crt file and the .key file into a new file. Save this new file as client-2048.pem. |

**Linking the Certificate to Your Betfair Account**

The previous steps should have created the following files:

|  |  |
| --- | --- |
| **File name** | **Description** |
| **client-2048.key** | The private key. This file is needed in order to use the certificate and should be protected and shouldn’t be shared with anyone. |
| **client-2048.csr** | A certificate signing request. This file is no longer needed and can be deleted. |
| **client-2048.crt** | The certificate. This file is not sensitive in security terms and can be shared with anyone. |

**Before you login using the certificate, it must be attached to your Betfair account, as follows:**

1. Log in to your Betfair account through betfair.comPaste the following URL into the address bar of your browser
2. Navigate to <https://myaccount.betfair.com/accountdetails/mysecurity?showAPI=1> **-  Note:**  Please use <https://myaccount.betfair.it/accountdetails/mysecurity?showAPI=1> for the **Italian Exchange.**
3. Scroll to the section at the bottom, titled “API-NG Configuration”
4. Click on “Browse” and then locate and select the file client-2048.crt (client-2048.pem if applicable) created above.
5. Click on the “Upload Certificate” button.

Scroll down to the “API-NG Configuration” section if required and the certificate details should be shown.  You should now be able to log in to your Betfair account using the API-NG endpoint.

**Note on File Formats**

Some systems require that client certificates are in a different format to the ones we’ve created.  The two most common formats are (a) PEM format key and certificate in a single file and (b) PKCS#12 format file.  .NET applications require a PKCS#12 format file.

**To create a PEM format file that contains both the private key and the certificate you can use the following command:**

**Linux**

|  |
| --- |
| cat client-2048.crt client-2048.key > client-2048.pem |

**Create the PKCS#12 format using crt and key**

|  |  |
| --- | --- |
| openssl pkcs12 -export -in client-2048.crt -inkey client-2048.key -out client-2048.p12 | |
| https://api.developer.betfair.com/services/webapps/docs/images/icons/emoticons/warning.png | Don't circulate the key, PEM file or PCKS#12 format files as these files are security sensitive |

**Details of a Login Request**

A login request can now be made as follows:

1. Submit a HTTP “POST” request to: <https://identitysso.betfair.com>[/api/certlogin](https://identitysso.betfair.com/api/certlogin)
2. As part of the SSL connection, the certificate created previously must be supplied.
3. Include a custom Header called “X-Application” with a value that identifies your application.  The value is not validated and is only used to help with troubleshooting and diagnosing any problems.
4. Ensure the POST’s Content-Type is “application/x-www-form-urlencoded” rather than MIME attachment encoded.
5. As part of the POST body include two parameters “username” and “password” which should have the relevant username/password for your account.

**Certificate Login Interface Details**

**URL Definition**

**Certificate Endpoint**

|  |
| --- |
| https://identitysso.betfair.com/api/certlogin |

*This endpoint is also available under:*

* identitysso.betfair.com
* identitysso.betfair.es
* identitysso.betfair.it
* identitysso.w-con.betfair.com
* identitysso.betfaironline.eu

**Request headers**

* **X-Application** - You must set the X-Application header to your application key.

**Request Parameters**

* **username** (mandatory) - The username of the user logging in.
* **password** (mandatory) - The password of the user logging in.

|  |  |
| --- | --- |
| https://api.developer.betfair.com/services/webapps/docs/images/icons/emoticons/warning.png | **Please note:**  The username and password values should be encoded when making the login request. All method names are case sensitive, this includes login, keepAlive and logout. |

**Response**

The response returned is a json string. If the response is successful then the loginStatus key will contain SUCCESS, for example:

|  |
| --- |
| {    sessionToken: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;    loginStatus: SUCCESS;  } |

Should a failure or exception be returned, the response will be structured as below and loginStatus will contain a failure reason:

|  |
| --- |
| {    loginStatus: INVALID\_USERNAME\_OR\_PASSWORD;  } |

The possible failure and exceptional return codes are:

| **loginStatus** |  |
| --- | --- |
| INVALID\_USERNAME\_OR\_PASSWORD | the username or password are invalid |
| ACCOUNT\_NOW\_LOCKED | the account was just locked |
| ACCOUNT\_ALREADY\_LOCKED | the account is already locked |
| PENDING\_AUTH | pending authentication |
| TELBET\_TERMS\_CONDITIONS\_NA | Telbet terms and conditions rejected |
| DUPLICATE\_CARDS | duplicate cards |
| SECURITY\_QUESTION\_WRONG\_3X | the user has entered wrong the security answer 3 times |
| KYC\_SUSPEND | KYC suspended |
| SUSPENDED | the account is suspended |
| CLOSED | the account is closed |
| SELF\_EXCLUDED | the account has been self-excluded |
| INVALID\_CONNECTIVITY\_TO\_REGULATOR\_DK | the DK regulator cannot be accessed due to some internal problems in the system behind or in at regulator; timeout cases included. |
| NOT\_AUTHORIZED\_BY\_REGULATOR\_DK | the user identified by the given credentials is not authorized in the DK's jurisdictions due to the regulators' policies. Ex: the user for which this session should be created is not allowed to act(play, bet) in the DK's jurisdiction. |
| INVALID\_CONNECTIVITY\_TO\_REGULATOR\_IT | the IT regulator cannot be accessed due to some internal problems in the system behind or in at regulator; timeout cases included. |
| NOT\_AUTHORIZED\_BY\_REGULATOR\_IT | the user identified by the given credentials is not authorized in the IT's jurisdictions due to the regulators' policies. Ex: the user for which this session should be created is not allowed to act(play, bet) in the IT's jurisdiction. |
| SECURITY\_RESTRICTED\_LOCATION | the account is restricted due to security concerns |
| BETTING\_RESTRICTED\_LOCATION | the account is accessed from a location where betting is restricted |
| TRADING\_MASTER | Trading Master Account |
| TRADING\_MASTER\_SUSPENDED | Suspended Trading Master Account |
| AGENT\_CLIENT\_MASTER | Agent Client Master |
| AGENT\_CLIENT\_MASTER\_SUSPENDED | Suspended Agent Client Master |
| DANISH\_AUTHORIZATION\_REQUIRED | Danish authorization required |
| SPAIN\_MIGRATION\_REQUIRED | Spain migration required |
| DENMARK\_MIGRATION\_REQUIRED | Denmark migration required |
| SPANISH\_TERMS\_ACCEPTANCE\_REQUIRED | The latest Spanish terms and conditions version must be accepted |
| ITALIAN\_CONTRACT\_ACCEPTANCE\_REQUIRED | The latest Italian contract version must be accepted |
| CERT\_AUTH\_REQUIRED | Certificate required or certificate present but could not authenticate with it |
| CHANGE\_PASSWORD\_REQUIRED | change password required |
| PERSONAL\_MESSAGE\_REQUIRED | personal message required for the user |
| INTERNATIONAL\_TERMS\_ACCEPTANCE\_REQUIRED | The latest international terms and conditions must be accepted prior to logging in. |

**Sample curl command to quickly check the certificate based login**

|  |
| --- |
| curl -q -k --cert client-2048.crt --key client-2048.key https://identitysso.betfair.com/api/certlogin -d "username=testuser&password=testpassword" -H "X-Application: curlCommandLineTest"    Response should be    {"sessionToken":"Zx8i4oigut5nc+l4L8qFb0DSxG+mwLn2t0AMGFxjrMJI=","loginStatus":"SUCCESS"} |

**Sample Code for Non-Interactive Login**

**Sample C# code using PKCS#12 key store**

Please see code sample via <https://github.com/betfair/API-NG-sample-code/tree/master/loginCode/Non-interactive-cSharp>

**Sample Java code using Apache http client library and  PKCS#12 key store**

**Java API-NG login**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85 | package com.test.aping.client;        import org.apache.http.HttpEntity;  import org.apache.http.HttpResponse;  import org.apache.http.NameValuePair;  import org.apache.http.client.entity.UrlEncodedFormEntity;  import org.apache.http.client.methods.HttpPost;  import org.apache.http.conn.ClientConnectionManager;  import org.apache.http.conn.scheme.Scheme;  import org.apache.http.conn.ssl.SSLSocketFactory;  import org.apache.http.conn.ssl.StrictHostnameVerifier;  import org.apache.http.impl.client.DefaultHttpClient;  import org.apache.http.message.BasicNameValuePair;  import org.apache.http.util.EntityUtils;    import javax.net.ssl.KeyManager;  import javax.net.ssl.KeyManagerFactory;  import javax.net.ssl.SSLContext;  import java.io.File;  import java.io.FileInputStream;  import java.io.InputStream;  import java.security.KeyStore;  import java.security.SecureRandom;  import java.util.ArrayList;  import java.util.List;      public class HttpClientSSO {        private static int port = 443;        public static void main(String[] args) throws Exception {            DefaultHttpClient httpClient = new DefaultHttpClient();            try {              SSLContext ctx = SSLContext.getInstance("TLS");              KeyManager[] keyManagers = getKeyManagers("pkcs12", new FileInputStream(new File("C:\\sslcerts\\client-2048.p12")), "password");              ctx.init(keyManagers, null, new SecureRandom());              SSLSocketFactory factory = new SSLSocketFactory(ctx, new StrictHostnameVerifier());                ClientConnectionManager manager = httpClient.getConnectionManager();              manager.getSchemeRegistry().register(new Scheme("https", port, factory));              HttpPost httpPost = new HttpPost("[https://identitysso.betfair.com/api/certlogin"](https://identitysso.betfair.com/api/certlogin));              List<NameValuePair> nvps = new ArrayList<NameValuePair>();              nvps.add(new BasicNameValuePair("username", "testuser"));              nvps.add(new BasicNameValuePair("password", "testpassword"));                httpPost.setEntity(new UrlEncodedFormEntity(nvps));                    httpPost.setHeader("X-Application","appkey");                  System.out.println("executing request" + httpPost.getRequestLine());                HttpResponse response = httpClient.execute(httpPost);              HttpEntity entity = response.getEntity();                System.out.println("----------------------------------------");              System.out.println(response.getStatusLine());              if (entity != null) {                  String responseString = EntityUtils.toString(entity);                  //extract the session token from responsestring                  System.out.println("responseString" + responseString);              }            } finally {              httpClient.getConnectionManager().shutdown();          }      }            private static KeyManager[] getKeyManagers(String keyStoreType, InputStream keyStoreFile, String keyStorePassword) throws Exception {          KeyStore keyStore = KeyStore.getInstance(keyStoreType);          keyStore.load(keyStoreFile, keyStorePassword.toCharArray());          KeyManagerFactory kmf = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());          kmf.init(keyStore, keyStorePassword.toCharArray());          return kmf.getKeyManagers();      }  } |

**Sample Python code**

|  |
| --- |
| #!/usr/bin/env python    import requests    #openssl x509 -x509toreq -in certificate.crt -out CSR.csr -signkey privateKey.key      payload = 'username=myusername&password=password'  headers = {'X-Application': 'SomeKey', 'Content-Type': 'application/x-www-form-urlencoded'}    resp = requests.post('[https://identitysso.betfair.com/api/certlogin'](https://identitysso.betfair.com/api/certlogin%27), data=payload, cert=('client-2048.crt', 'client-2048.key'), headers=headers)    if resp.status\_code == 200:    resp\_json = resp.json()    print resp\_json['loginStatus']    print resp\_json['sessionToken']  else:    print "Request failed." |