**Get redirection URL from UnityWebRequest**

I am issuing a UnityWebRequest with a URL that has redirection involved. The redirection is being logged as a warning by the compiler. Is there any way of accessing that redirection URL?

I've tried the method of using ResponseHeaders["Location"]. That does not work as the Response Headers don't have such a field.

Can someone help me log that warning URL? I don't want to write a dump of the console into a file and parse it from there. Is there a cleaner approach?

**0**

**Answer** by [gizmhail](http://answers.unity3d.com/users/874377/gizmhail.html) · Apr 09 at 05:35 AM

I also tried with UnityWebRequest and WWW to find the Location header in the responses, and I didn't find a proper way to do so.

So, I ended using directly TcpClient, and it kinda works. *Remark: it looks like my implementation is not very efficient/stable, as you'll notice that the request take an usual time and sometimes fail. I'll investigate it later.*

If you simply want to reach an http server, with a GET request, it is pretty straighforward.

If you want to POST a form, you have to build the request body.

And if you want to use an https connection, it becomes a bit tricky :

1. you have to use an SslStream
2. as certificates don't seem to be properly loaded, you have to handle their manipulation manually
3. I still haven't found how to do it securely, so for the moment, I **completely disable the certificate check. Please keep in mind that it is not safe at all**.

So, first, for the simple case (an http GET request, whose we want to find the redirection) :

1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using System.Net.Sockets;
5. using System.IO;
7. public class TCPTest : MonoBehaviour {
8. public string redirection = null;
9. public string result = null;
10. public bool connected = false;
12. // Use this for initialization
13. void Start () {
14. fetchGetHttpRedirection ();
15. }
17. // Update is called once per frame
18. void Update () {
19. if (!this.connected) {
20. if (this.redirection != null) {
21. Debug.Log ("Redirection:\n" + this.redirection);
22. this.redirection = null;
23. } else if(this.result != null) {
24. Debug.Log ("Answer:\n" + this.result);
25. this.result = null;
26. }
27. }
28. }
30. void fetchGetHttpRedirection(){
31. StartCoroutine (GetTCPClientRedirect ("example.com","/"));
33. }
35. IEnumerator GetTCPClientRedirect(string host, string query){
36. TcpClient client = new TcpClient(host, 80);
37. NetworkStream networkStream = client.GetStream();
38. StreamReader reader = new StreamReader(networkStream);
39. StreamWriter writer = new StreamWriter(networkStream);
40. // Basic GET request:
41. // GET /foo.html HTTP/1.1\r\nHost: example.com\r\n
42. writer.WriteLine("GET "+ query + " HTTP/1.1");
43. writer.WriteLine("Host: " + host);
44. writer.WriteLine("");
45. writer.Flush ();
46. this.redirection = null; // variable where we'll store the redirection
47. string lastLine = null;
48. this.result = "";
49. this.connected = true;
50. while (true) {
51. if (!client.Connected) {
52. break;
53. }
55. if (networkStream.CanRead) {
56. string line = reader.ReadLine ();
57. if (line != null) {
58. string redirectionHeader = "Location: ";
59. if(line.StartsWith(redirectionHeader)){
60. this.redirection = line.Substring (redirectionHeader.Length, line.Length - redirectionHeader.Length);
61. }
62. if (this.result != "") {
63. this.result = this. result + "\n";
64. }
66. if (lastLine == "" && line == "0") {
67. break;
68. } else {
69. lastLine = line;
70. this.result = this.result + line;
71. }
72. } else {
73. break;
74. }
75. }
76. // To avoid blocking the thread
77. yield return null;
78. }
79. // Disconnection
80. networkStream.Close();
81. client.Close();
82. this.connected = false;
83. yield return null;
84. }
86. }

Then, an example with both a POST and the (unsecure) https handling :

1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using System.Net.Sockets;
5. using System.IO;
6. using UnityEngine.Networking;
7. using System.Net.Security;
8. using System.Security.Cryptography.X509Certificates;
9. using System.Security.Authentication;
11. public class TCPPostSSLTest:MonoBehaviour
12. {
13. SampleAuthentificationRequest request = null;
15. void Start () {
16. string host = "example.com";
17. string authorizationQuery = "/foo/login.html";
18. request = new SampleAuthentificationRequest (host, authorizationQuery);
19. StartCoroutine(request.Launch ("<login>", "<password>"));
20. }
22. // Update is called once per frame
23. void Update () {
24. if (!this.request.connected) {
25. if (this.request.redirection != null) {
26. Debug.Log ("Redirection:\n" + this.request.redirection);
27. this.request.redirection = null;
28. } else if(this.request.result != null) {
29. Debug.Log ("Answer:\n" + this.request.result);
30. this.request.result = null;
31. }
32. }
33. }
35. public class SampleAuthentificationRequest
36. {
37. public string result;
38. public string redirection = null;
39. public bool connected = false;
40. string host;
41. string authorizationQuery;
42. public SampleAuthentificationRequest(string host, string authorizationQuery){
43. this.host = host;
44. this.authorizationQuery = authorizationQuery;
46. }
48. public static bool CertificateValidationCallback(
49. object sender,
50. X509Certificate certificate,
51. X509Chain chain,
52. SslPolicyErrors sslPolicyErrors)
53. {
54. Debug.Log ("[Warning] Using unsafe certification: always accepting server certificats !");
55. return true;
56. }
58. public IEnumerator Launch(string username, string password) {
59. string body = "username=" + WWW.EscapeURL(username) + "&password=" + WWW.EscapeURL(password) + "&login=1";
60. int contentLength = System.Text.Encoding.UTF8.GetBytes(body).Length;
62. TcpClient client = new TcpClient(this.host, 443);
63. NetworkStream networkStream = client.GetStream();
64. SslStream sslStream = new SslStream(networkStream
65. ,false
66. ,new RemoteCertificateValidationCallback(CertificateValidationCallback)
67. );
68. // Debug.Log("Authenticating...");
69. sslStream.AuthenticateAsClient (host);
70. // Debug.Log("Authent done");
71. while (!sslStream.IsAuthenticated) {
72. // Debug.Log("Not yet authenticated...");
73. yield return null;
74. }
75. this.connected = true;
76. StreamReader reader = new StreamReader(sslStream);
77. StreamWriter writer = new StreamWriter(sslStream);
79. string requestData = "POST " + this.authorizationQuery + " HTTP/1.1\r\n";
80. requestData = requestData + "Host: " + this.host + "\r\n";
81. requestData = requestData + "User-Agent: UniNoco\r\n";
82. requestData = requestData + "Accept: \*/\*\r\n";
83. requestData = requestData + "Content-Length: " + contentLength + "\r\n";
84. requestData = requestData + "Content-Type: application/x-www-form-urlencoded\r\n";
85. requestData = requestData + "\r\n";
86. requestData = requestData + body + "\r\n";
87. requestData = requestData + "\r\n";
88. requestData = requestData + "\r\n";
89. writer.Write (requestData);
90. writer.Flush ();
91. // Debug.Log ("Request:");
92. Debug.Log(requestData);
93. result = "";
94. string lastLine = null;
95. while (true) {
96. if (!client.Connected) {
97. // Debug.Log ("Disconnected");
98. break;
99. }
101. if (sslStream.CanRead) {
102. string line = reader.ReadLine ();
103. if (line != null) {
104. string redirectionHeader = "Location: ";
105. if(line.StartsWith(redirectionHeader)){
106. this.redirection = line.Substring (redirectionHeader.Length, line.Length - redirectionHeader.Length);
107. }
108. if (result != "") {
109. result = result + "\n";
110. }
112. if (lastLine == "" && line == "0") {
113. break;
114. }
115. lastLine = line;
117. // Debug.Log ("Received: >" + line + "<");
118. result = result + line;
119. } else {
120. break;
121. }
122. }
123. // Debug.Log ("No data ...");
124. yield return null;
126. }
128. // Debug.Log("Closing...");
129. sslStream.Close();
130. client.Close();
131. // Debug.Log("Closed.");
132. this.connected = false;
133. yield return null;
134. }
135. }
136. }

[Add comment](http://answers.unity3d.com/questions/1326554/get-redirection-url-from-unitywebrequest.html) · [Share](http://answers.unity3d.com/questions/1326554/get-redirection-url-from-unitywebrequest.html)

[avatar image](http://answers.unity3d.com/users/6612/bunny83.html)

**0**

**Answer** by [Bunny83](http://answers.unity3d.com/users/6612/bunny83.html) · Apr 09 at 03:02 AM

Unfortunately no. The WWW class as well as the UnityWebRequest class follow a redirection response transparently. So they don't provide any information about that redirect. The UnityWebRequest has a [redirectLimit](https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest-redirectLimit.html), however it doesn't provide informations of the redirection but only generates an error if the redirection amount exceeds that limit.

I never was in need of getting the redirection URL manually. However if the URL is dumped as warning you could intercept it by using a [log-callback](https://docs.unity3d.com/ScriptReference/Application-logMessageReceived.html).

A cleaner approach would be to use a more low-level API. The .NET / Mono WebRequest class does [allow to disable the auto-redirect](http://stackoverflow.com/questions/14012384/webrequest-prevent-redirection). That way you get the actual response and you have to follow the new URL manually.

[Add comment](http://answers.unity3d.com/questions/1326554/get-redirection-url-from-unitywebrequest.html) ·  Hide 3 · [Share](http://answers.unity3d.com/questions/1326554/get-redirection-url-from-unitywebrequest.html)

[avatar imagegizmhail](http://answers.unity3d.com/users/874377/gizmhail.html)· [Apr 09 at 12:29 PM](http://answers.unity3d.com/comments/1337583/view.html) 0

You're right, it is also possible to do it with WebRequest (and it is simpler to write, thank you ^^)

However, I still have my certificate validation problem with it, and here, I have to desactive safe validation globally, not just for the request. I tried to find a better custom validator (see the referenced [answers.unity3d.com](http://answers.unity3d.com/) question for details), but I'm still not sure that this solution can be considered safe.

Anyway, even if unsecure, it "works" this way too :) :

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using System.Net;

using System.Net.Security;

using System.IO;

using System.Security.Cryptography.X509Certificates;

public class HttpWebRequestTest : MonoBehaviour {

// Use this for initialization

void Start () {

StartCoroutine(SSLPostRequest ("<login>", "<password>"));

}

// Update is called once per frame

void Update () {

}

// Source: http://answers.unity3d.com/questions/50013/httpwebrequestgetrequeststream-https-certificate-e.html

public bool CustomCertificateValidationCallback(System.Object sender, X509Certificate certificate, X509Chain chain, SslPolicyErrors sslPolicyErrors) {

bool isOk = true;

// If there are errors in the certificate chain, look at each error to determine the cause.

if (sslPolicyErrors != SslPolicyErrors.None) {

for (int i=0; i<chain.ChainStatus.Length; i++) {

if (chain.ChainStatus [i].Status != X509ChainStatusFlags.RevocationStatusUnknown) {

chain.ChainPolicy.RevocationFlag = X509RevocationFlag.EntireChain;

chain.ChainPolicy.RevocationMode = X509RevocationMode.Online;

chain.ChainPolicy.UrlRetrievalTimeout = new TimeSpan (0, 1, 0);

chain.ChainPolicy.VerificationFlags = X509VerificationFlags.AllFlags;

bool chainIsValid = chain.Build ((X509Certificate2)certificate);

if (!chainIsValid) {

isOk = false;

}

}

}

}

return isOk;

}

IEnumerator SSLPostRequest(string username, string password){

RemoteCertificateValidationCallback defaultCertificateValidationCallback = ServicePointManager.ServerCertificateValidationCallback;

ServicePointManager.ServerCertificateValidationCallback = CustomCertificateValidationCallback;

string body = "username=" + WWW.EscapeURL(username) + "&password=" + WWW.EscapeURL(password) + "&login=1";

string host = "example.com";

string authorizationQuery = "/authentificationWithRedirect.html";

var request = (HttpWebRequest)WebRequest.Create("https://"+host+authorizationQuery);

request.AllowAutoRedirect = false;

var data = System.Text.Encoding.UTF8.GetBytes(body);

request.Method = "POST";

request.ContentType = "application/x-www-form-urlencoded";

request.ContentLength = data.Length;

using (var stream = request.GetRequestStream())

{

Debug.Log ("Sending:\n" + data);

stream.Write(data, 0, data.Length);

}

var response = (HttpWebResponse)request.GetResponse();

Debug.Log ("Location header: " + response.GetResponseHeader("Location"));

var responseString = new StreamReader(response.GetResponseStream()).ReadToEnd();

Debug.Log ("response: "+ responseString);

ServicePointManager.ServerCertificateValidationCallback = defaultCertificateValidationCallback;

yield return null;

}

}

[avatar imageBunny83](http://answers.unity3d.com/users/6612/bunny83.html) gizmhail · [Apr 09 at 02:05 PM](http://answers.unity3d.com/comments/1337604/view.html) 0

Certificates should be validated automatically. However if you have an unregistrated certificate you have to do the validation yourself. Keep in mind that SSL build on top of [the chain of trust](https://en.wikipedia.org/wiki/Chain_of_trust#Computer%5Fsecurity). If you use a certificate that is not registrated at a [certificate authority](https://en.wikipedia.org/wiki/Certificate_authority) a client can never validate the certificate "securely". This is the case when a webbrowser requires you to add a manual exception to simply accept the certificate locally. You could ship your public certificate with your build so you may be able to simply compare them.

ps: you did not URL encode your username and password in your post data. This can cause problems when those contain restricted characters (like "=", "%", "&", ...)

[avatar imagegizmhail](http://answers.unity3d.com/users/874377/gizmhail.html) Bunny83 · [Apr 13 at 09:07 PM](http://answers.unity3d.com/comments/1340428/view.html) 0

[Share](http://answers.unity3d.com/questions/1326554/get-redirection-url-from-unitywebrequest.html)

Thanks, you're right about escaping the params ! I updated my snippets to add the escaping

Concerning the certificate of the API I'm connecting too, it seems valid (based on Let's Encrypt). I think that the certificate store is not loaded when using TcpClient or HttpWebRequest: I should find a way to manually trigger the store loading, but I've not yet found it.

Anyway, thanks for the advices :)