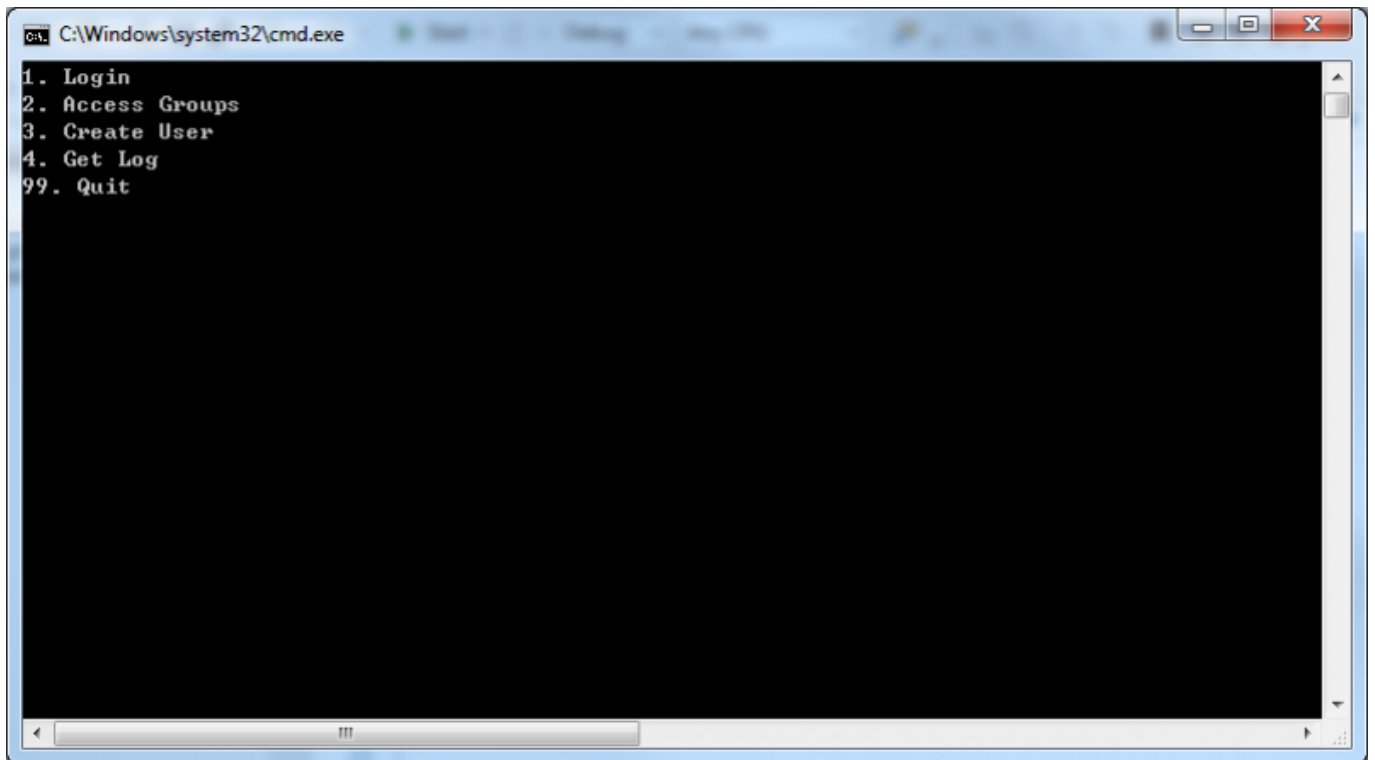


BioStar 2 API 가 .....	1
.....	1
.....	1
.....	4
.....	8

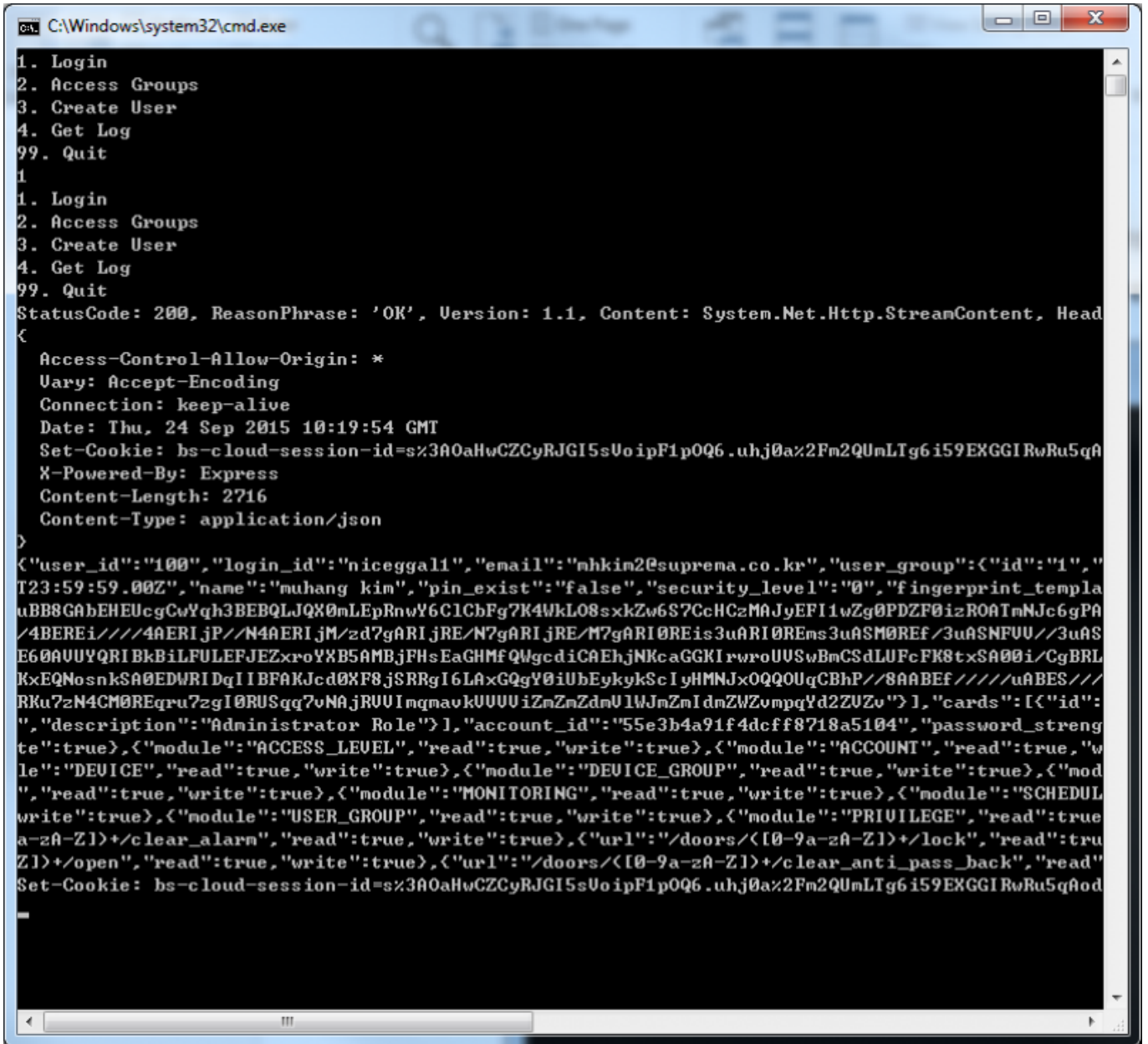
# BioStar 2 API 가

BioStar API <https://api.biostar2.com> . C#  
 BioStar API .  
 Windows .  
 BioStar API RESTful API RESTful API BioStar API .

가 , Visual C# , 가 ,  
 가 , 가 .



[ 1. ]  
 BioStar . '1' <Enter>  
 BioStar . BioStar API BioStar  
 , BioStar BioStar API BioStar BioStar API  
 BioStar API 가



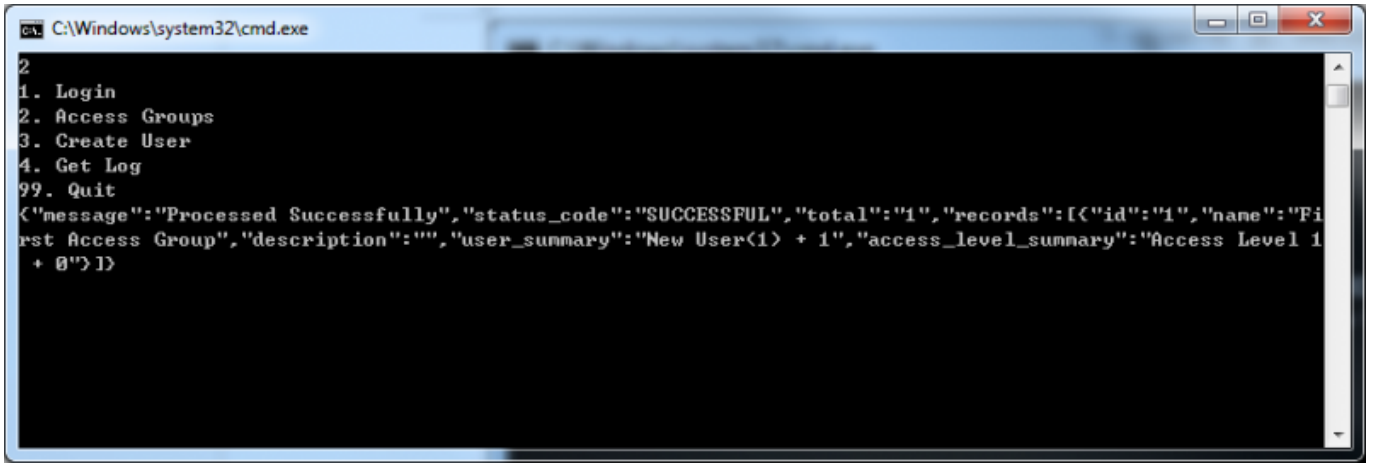
[ 2. ]

가 . JSON

, 2

가 가 "2"

<Enter>



[ 3. 가 ]

BioStar 2 "First Access Group" . "user\_summary" 가 " " 3" <Enter> . ID 가 ID "98" 가 .



[ 4. ]

가 . "4" Enter  
가 .

```

4
1. Login
2. Access Groups
3. Create User
4. Get Log
99. Quit
Succeeded to retrieve log from 2015-09-21T10:07:27Z to 2015-09-24T10:45:28Z
{"message": "Processed Successfully", "status_code": "SUCCESSFUL", "total": 0, "records": []}
Succeeded to retrieve log from 1970-01-01T00:00:00Z to 2015-09-24T10:45:28Z
{"message": "Processed Successfully", "status_code": "SUCCESSFUL", "total": "3437", "records": [{"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T10:07:26.00Z", "id": "68002", "index": "341", "server_datetime": "2015-09-21T19:07:26.00Z", "user": {"user_id": "56"}, "event_type": {"code": "9216", "name": "DELETE_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "DELETE_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T10:06:08.00Z", "id": "6801", "index": "340", "server_datetime": "2015-09-21T19:07:08.00Z", "user": {"user_id": "56"}, "event_type": {"code": "8192", "name": "ENROLL_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "ENROLL_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T10:01:17.00Z", "id": "6800", "index": "339", "server_datetime": "2015-09-21T19:01:18.00Z", "user": {"user_id": "33"}, "event_type": {"code": "9216", "name": "DELETE_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "DELETE_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T10:00:06.00Z", "id": "6799", "index": "338", "server_datetime": "2015-09-21T19:01:08.00Z", "user": {"user_id": "33"}, "event_type": {"code": "8192", "name": "ENROLL_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "ENROLL_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T09:57:50.00Z", "id": "6797", "index": "337", "server_datetime": "2015-09-21T18:57:51.00Z", "user": {"user_id": "33"}, "event_type": {"code": "9216", "name": "DELETE_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "DELETE_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T09:57:17.00Z", "id": "6798", "index": "336", "server_datetime": "2015-09-21T18:58:18.00Z", "user": {"user_id": "33"}, "event_type": {"code": "8192", "name": "ENROLL_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "ENROLL_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T09:47:00.00Z", "id": "6795", "index": "335", "server_datetime": "2015-09-21T18:47:01.00Z", "user": {"user_id": "33"}, "event_type": {"code": "9216", "name": "DELETE_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "DELETE_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T09:46:02.00Z", "id": "6796", "index": "334", "server_datetime": "2015-09-21T18:47:04.00Z", "user": {"user_id": "33"}, "event_type": {"code": "8192", "name": "ENROLL_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "ENROLL_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T09:44:44.00Z", "id": "6793", "index": "333", "server_datetime": "2015-09-21T18:44:46.00Z", "user": {"user_id": "33"}, "event_type": {"code": "9216", "name": "DELETE_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "DELETE_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T09:43:53.00Z", "id": "6794", "index": "332", "server_datetime": "2015-09-21T18:44:54.00Z", "user": {"user_id": "33"}, "event_type": {"code": "8192", "name": "ENROLL_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "ENROLL_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T09:42:11.00Z", "id": "6792", "index": "331", "server_datetime": "2015-09-21T18:42:12.00Z", "user": {"user_id": "33"}, "event_type": {"code": "9216", "name": "DELETE_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "DELETE_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T08:06:53.00Z", "id": "6791", "index": "330", "server_datetime": "2015-09-21T17:07:54.00Z", "user": {"user_id": "33"}, "event_type": {"code": "8704", "name": "UPDATE_SUCCESS", "alertable": "false", "enable_alert": "false", "description": "UPDATE_SUCCESS"}, "type": "USER", "level": "GREEN"}, {"device": {"id": "546833022", "name": "BioStation 2 546833022 <192.168.16.158>"}, "datetime": "2015-09-21T05:57:53.00Z", "id": "6790", "index": "329", "server_datetime": "2015-09-21T14:58:54.00Z", "user": {"user_id": "33"}, "event_type": {"code": "8704", "name": "UPDATE_SUCCESS", "alertable": "fal

```

[ 5. 가 ]

1.

가

```

24     static async void LoginTask()
25     {
26         string resourceAddress = "https://api.biostar2.com/v1/login";
27
28         HttpClient httpClient = new HttpClient();
29
30         JavaScriptSerializer serializer = new JavaScriptSerializer();
31
32         Dictionary<string, string> dicLoginUser = new Dictionary<string, string>();
33         dicLoginUser.Add("name", "ts22");
34         dicLoginUser.Add("password", "rlaangkd!1");
35         dicLoginUser.Add("user_id", "niceggall");
36
37         string jsonLoginUser = serializer.Serialize(dicLoginUser);
38
39         StringContent sc = new StringContent(jsonLoginUser, Encoding.UTF8, "application/json");
40         HttpResponseMessage httpResponse = await httpClient.PostAsync(resourceAddress, sc);
41
42
43         if(httpResponse.IsSuccessStatusCode == true)
44         {
45             Console.WriteLine(httpResponse.ToString());
46             string httpResponseBody = await httpResponse.Content.ReadAsStringAsync();
47             Console.WriteLine(httpResponseBody);
48
49
50             MemoryStream responseMemoryStream = new MemoryStream();
51             StreamWriter sw = new StreamWriter(responseMemoryStream);
52             sw.Write(httpResponse.ToString());
53             sw.Flush();
54
55             bool isSessionIDContained = httpResponse.Headers.Contains("Set-Cookie");
56             if (isSessionIDContained == true)
57             {
58                 IEnumerable<string> sessionEnum = httpResponse.Headers.GetValues("Set-Cookie");
59                 foreach(string element in sessionEnum)
60                 {
61                     Console.WriteLine("Set-Cookie: " + element);
62                     string[] strCookieArr = element.Split(new string[] { "bs-cloud-session-id=" }, StringSplitOptions.None);
63                     string[] strCookieArr2 = strCookieArr[1].Split(new string[] { ";" }, StringSplitOptions.None);
64                     sessionID = strCookieArr2[0];
65                 }
66             }
67             else
68             {
69                 Console.WriteLine("Session ID not found");
70             }
71         }
72         else
73         {
74             Console.WriteLine("Failed to log in");
75             Console.WriteLine(httpResponse.ToString());
76         }
77     }

```

- 26 : BioStar URL . HTTPS  
"api.biostar2.com/v1/" "login"
- 27 : BioStar HttpClient
- 30 : JSON 가 JSON  
JavaScriptSerializer 가
- 32~35 : , ID, ,  
가 가 "name" 가
- 37 : JSON
- 39 : JSON HTTP , UTF8 , JSON
- 40 : HTTP HTTP POST
- 45~53 : HTTP
- 55~65 : 가 가 . API HTTP  
, 55~65 HTTP

## 2. 가

```

203 static async void AccessGroupsTask()
204 {
205     if (sessionID == null)
206     {
207         Console.WriteLine("You must log in first!");
208         return;
209     }
210
211     CookieContainer cookieContainer = new CookieContainer();
212
213     HttpClientHandler handler = new HttpClientHandler();
214     handler.CookieContainer = cookieContainer;
215
216     HttpClient client = new HttpClient(handler);
217
218
219     cookieContainer.Add(new Uri("https://api.biostar2.com"), new Cookie("bs-cloud-session-id", sessionID));
220     HttpResponseMessage httpResponse = await client.GetAsync("https://api.biostar2.com/v1/access_groups");
221
222     if (httpResponse.IsSuccessStatusCode == true)
223     {
224         string httpResponseBody = await httpResponse.Content.ReadAsStringAsync();
225         Console.WriteLine(httpResponseBody);
226     }
227     else
228     {
229         Console.WriteLine("Retrieving Access Groups Failed");
230         Console.WriteLine(httpResponse.ToString());
231     }
232 }

```

- 205~209 : ID가
- 211 : BioStar ID CookieContainer
- 219 : ID URI 가
- 220 : HTTP GET 가

## 3. 가

```

125     static async void GetLogTask()
126     {
127         if(sessionID == null)
128         {
129             Console.WriteLine("You must log in first!");
130             return;
131         }
132
133         CookieContainer cookieContainer = new CookieContainer();
134
135         HttpClientHandler handler = new HttpClientHandler();
136         handler.CookieContainer = cookieContainer;
137
138         HttpClient httpClient = new HttpClient(handler);
139
140         HttpClient client = new HttpClient(handler);
141         cookieContainer.Add(new Uri("https://api.biostar2.com"), new Cookie("bs-cloud-session-id", sessionID));
142
143         string resourceAddress = "https://api.biostar2.com/v1/monitoring/event_log/search";
144
145         string startTime = "1970-01-01T00:00:00Z";
146         string endTime = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ");
147
148         DateTime dtLatestLogTime = new DateTime(1970, 1, 1);
149
150         JavaScriptSerializer serializer = new JavaScriptSerializer();
151
152         for (int logCallIndex = 0; logCallIndex < 1000; logCallIndex++)
153         {
154             endTime = DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ssZ");
155
156             string payload = "{ \"datetime\": [\"" + startTime + "\", \"" + endTime + "\"]}";
157
158             StringContent sc = new StringContent(payload, Encoding.UTF8, "application/json");
159             HttpResponseMessage httpResponse = await httpClient.PostAsync(resourceAddress, sc);
160
161             if (httpResponse.IsSuccessStatusCode == true)
162             {
163                 Console.WriteLine("Succeeded to retrieve log from " + startTime + " to " + endTime);
164                 string httpResponseBody = await httpResponse.Content.ReadAsStringAsync();
165                 Console.WriteLine(httpResponseBody);
166
167                 endTime = startTime;
168
169                 Dictionary<string, dynamic> logValues = serializer.Deserialize<Dictionary<string, dynamic>>(httpResponseBody);
170                 foreach(KeyValuePair<string, dynamic> logElement in logValues)
171                 {
172                     if (logElement.Key == "records")
173                     {
174                         foreach (Dictionary<string, dynamic> recordElement in logElement.Value)
175                         {
176                             if(recordElement.ContainsKey("datetime"))
177                             {
178                                 Console.WriteLine(recordElement["datetime"]);
179                                 DateTime dtLogTime = DateTime.Parse(recordElement["datetime"]);
180
181                                 if(dtLogTime > dtLatestLogTime)
182                                 {
183                                     dtLatestLogTime = dtLogTime;
184                                     startTime = dtLatestLogTime.ToUniversalTime().AddSeconds(1).ToString("yyyy-MM-ddTHH:mm:ssZ");
185                                 }
186                             }
187                         }
188                     }
189                 }
190
191                 System.Threading.Thread.Sleep(1000);
192             }
193             else
194             {
195                 Console.WriteLine("Log Retrieval Failed from " + startTime + " to " + endTime);
196                 Console.WriteLine(httpResponse.ToString());
197                 break;
198             }
199         }

```

- 152 : 가 For
- 154~156 : 가 , JSON
- 170~188 : JSON Dictionary



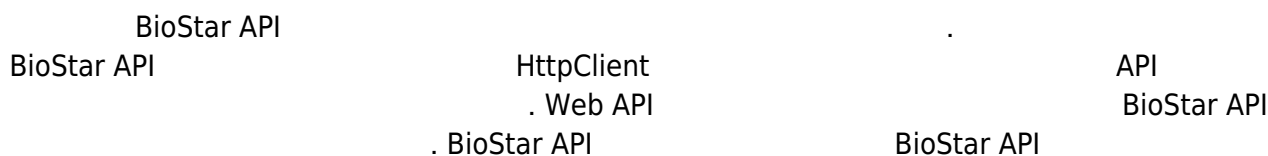
4.

```

79 static async void CreateUserTask()
80 {
81     if (sessionID == null)
82     {
83         Console.WriteLine("You must log in first!");
84         return;
85     }
86
87     CookieContainer cookieContainer = new CookieContainer();
88
89     HttpClientHandler handler = new HttpClientHandler();
90     handler.CookieContainer = cookieContainer;
91
92     HttpClient httpClient = new HttpClient(handler);
93
94     HttpClient client = new HttpClient(handler);
95     cookieContainer.Add(new Uri("https://api.biostar2.com"), new Cookie("bs-cloud-session-id", sessionID));
96
97     string resourceAddress = "https://api.biostar2.com/v1/users";
98
99     Console.WriteLine("Input User ID: ");
100    string userInputID = Console.ReadLine();
101
102    JavaScriptSerializer serializer = new JavaScriptSerializer();
103
104    Dictionary<string, string> dicNewUser = new Dictionary<string, string>();
105    dicNewUser.Add("user_id", userInputID);
106
107    string payload = serializer.Serialize(dicNewUser);
108
109    StringContent sc = new StringContent(payload, Encoding.UTF8, "application/json");
110    HttpResponseMessage httpResponse = await httpClient.PostAsync(resourceAddress, sc);

```

- 99~100 : ID
- 104~105 : ID



<https://api.biostar2.com>

From: <https://kb.supremainc.com/knowledge/> -

Permanent link: [https://kb.supremainc.com/knowledge/doku.php?id=ko:biostar\\_2\\_api\\_quickstart\\_guide](https://kb.supremainc.com/knowledge/doku.php?id=ko:biostar_2_api_quickstart_guide)

Last update: **2016/12/30 11:14**