

```
1 using System;
2 using System.Diagnostics;
3 using System.Net.Sockets;
4
5 namespace SmokeScreen.Modules
6 {
7     /// <summary>
8     /// Interface For Logging Results of Client/Server Operations...
9     /// </summary>
10    public static class Logging
11    {
12
13        /// <summary>
14        /// Client Generated Log Messages
15        /// </summary>
16        public static class Client
17        {
18            public static class Log
19            {
20                public static void ResponseRecieved(string response)
21                {
22                    Console.WriteLine($"Response received : {response}");
23                    Debug.WriteLine($"Response received : {response}");
24                }
25
26                public static void ConnectionSuccess(Socket client)
27                {
28                    Console.WriteLine($"Socket connected to {client.RemoteEndPoint.ToString()}");
29                    Debug.WriteLine($"Socket connected to {client.RemoteEndPoint.ToString()}");
30                }
31
32                public static void Issue(Exception exception)
33                {
34                    Console.WriteLine($"{exception}");
35                    Debug.WriteLine($"{exception}");
36                }
37
38                public static void ByteCount(int length)
39                {
40                    Console.WriteLine($"Sent {length} bytes to client.");
41                    Debug.WriteLine($"Sent {length} bytes to client.");
42                }
43            }
44        }
45
46        /// <summary>
47        /// Server Generated Log Messages
48        /// </summary>
49        public static class Server
50        {
```

```
51     public static class Log
52     {
53         public static void ConnectionReady()
54         {
55             Console.WriteLine("Waiting for a Connection...");
56             Debug.WriteLine("Waiting for a Connection...");
57         }
58
59         public static void BytesRead(string transaction, int length)
60         {
61             Console.WriteLine($"Read {length} bytes from socket. \n      ↗
62             Data : {transaction}");
63             Debug.WriteLine($"Read {length} bytes from socket. \n Data : ↗
64             {transaction}");
65         }
66
67         public static void ByteCount(int length)
68         {
69             Console.WriteLine($"Sent {length} bytes to client.");
70             Debug.WriteLine($"Sent {length} bytes to client.");
71         }
72
73         public static void MessageRecieved(string message)
74         {
75             Console.WriteLine($"Recieved '{message}' from client");
76             Debug.WriteLine($"Recieved '{message}' from client");
77         }
78
79         public static void InvalidKey(string description = "")
80         {
81             Console.WriteLine($"Invalid Key Recieved {description}");
82             Debug.WriteLine($"Invalid Key Recieved {description}");
83         }
84
85         public static void InvalidMessageFormat(string description = "")
86         {
87             Console.WriteLine($"Invalid Message Format Recieved      ↗
88             {description}");
89             Debug.WriteLine($"Invalid Message Format Recieved      ↗
90             {description}");
91         }
92
93         public static void UnauthorizedRequest(string description = "")
94         {
95             Console.WriteLine($"Unauthorized Request from      ↗
96             {description}");
97             Debug.WriteLine($"Unauthorized Request from {description}");
98         }
99
100        public static void Issue(Exception exception)
101        {
102            Console.WriteLine(exception.ToString());
103        }
104    }
```

```
198         Debug.WriteLine(exception.ToString());
199     }
200
201     public static void Decryption(string result)
202     {
203         Console.WriteLine($"Decryption: {result}");
204         Debug.WriteLine($"Decryption: {result}");
205     }
206
207     public static void ProcessKey(bool truth, string clientPublicKey ↗
208         = "")
209     {
210         if (truth)
211         {
212             Console.WriteLine($"Successfully Authenticated Key: ↗
213             {clientPublicKey}");
214             Debug.WriteLine($"Successfully Authenticated Key: ↗
215             {clientPublicKey}");
216         }
217         else
218         {
219             Console.WriteLine($"Warning unable to authenticate key. ↗
220             The ring was not updated.");
221             Debug.WriteLine($"Warning unable to authenticate key. The ↗
222             ring was not updated.");
223         }
224     }
225 }
226
227 /*
228 /// <summary>
229 /// Helper Class for Giving Debug.Write Functionality to other ↗
230 Resources..
231 /// </summary>
232 public static class Helper
233 {
234     public static void Write(string message)
235     {
236         Debug.WriteLine(message);
237     }
238
239     public static void Write(params object[] list)
240     {
241         if (list.Length == 0)
242         {
243             Debug.WriteLine();
244         }
245         else if (list.Length == 1)
246         {
247             Debug.WriteLine(string.Format("{0}", list[0]));
248         }
249     }
250 }
```

```
144         else
145         {
146             Debug.WriteLine(string.Format(list[0].ToString(), GetArgs    ↗
147                                     (list)));
148         }
149     }
150
151     /// <summary>
152     /// From Array Extracts 2nd..Last
153     /// </summary>
154     private static object[] GetArgs(object[] list)
155     {
156         int argCount = list.Length - 1;
157         object[] args = new object[argCount];
158         for (int i = 0; i < argCount; i++)
159         {
160             args[i] = list[i + 1];
161         }
162         return args;
163     }
164 }
165 */
166
167 }
168 }
169
```