```
using System;
using System.Diagnostics;
using System.Net.Sockets;

namespace SmokeScreen.Modules
{
    /// <summary>
    /// Interface For Logging Results of Client/Server Operations...
    /// </summary>
    public static class Logging
    {

        /// <summary>
        /// Client Generated Log Messages
        /// </summary>
        public static class Client
        {
            public static class Log
            {
                public static void ResponseRecieved(string response)
                {
                    Console.WriteLine($"Response received : {response}");
                    Debug.WriteLine($"Response received : {response}");
                }

                public static void ConnectionSuccess(Socket client)
                {
                    Console.WriteLine($"Socket connected to 
                     {client.RemoteEndPoint.ToString()}");
                    Debug.WriteLine($"Socket connected to 
                     {client.RemoteEndPoint.ToString()}");
                }

                public static void Issue(Exception exception)
                {
                    Console.WriteLine($"{exception}");
                    Debug.WriteLine($"{exception}");
                }

                public static void ByteCount(int length)
                {
                    Console.WriteLine($"Sent {length} bytes to client.");
                    Debug.WriteLine($"Sent {length} bytes to client.");
                }
            }
        }

        /// <summary>
        /// Server Generated Log Messages
        /// </summary>
        public static class Server
        {
```

```csharp
        public static class Log
        {
            public static void ConnectionReady()
            {
                Console.WriteLine("Waiting for a Connection...");
                Debug.WriteLine("Waiting for a Connection...");
            }

            public static void BytesRead(string transaction, int length)
            {
                Console.WriteLine($"Read {length} bytes from socket. \n Data : {transaction}");
                Debug.WriteLine($"Read {length} bytes from socket. \n Data : {transaction}");
            }

            public static void ByteCount(int length)
            {
                Console.WriteLine($"Sent {length} bytes to client.");
                Debug.WriteLine($"Sent {length} bytes to client.");
            }

            public static void MessageRecieved(string message)
            {
                Console.WriteLine($"Recieved '{message}' from client");
                Debug.WriteLine($"Recieved '{message}' from client");
            }

            public static void InvalidKey(string description = "")
            {
                Console.WriteLine($"Invalid Key Recieved {description}");
                Debug.WriteLine($"Invalid Key Recieved {description}");
            }

            public static void InvalidMessageFormat(string description = "")
            {
                Console.WriteLine($"Invalid Message Format Recieved {description}");
                Debug.WriteLine($"Invalid Message Format Recieved {description}");
            }

            public static void UnAuthorizedRequest(string description = "")
            {
                Console.WriteLine($"Unauthorized Request from {description}");
                Debug.WriteLine($"Unauthorized Request from {description}");
            }

            public static void Issue(Exception exception)
            {
                Console.WriteLine(exception.ToString());
```

```
                    Debug.WriteLine(exception.ToString());
                }

                public static void Decryption(string result)
                {
                    Console.WriteLine($"Decryption: {result}");
                    Debug.WriteLine($"Decryption: {result}");
                }

                public static void ProcessKey(bool truth, string clientPublicKey = "")
                {
                    if (truth)
                    {
                        Console.WriteLine($"Successfully Authenticated Key: {clientPublicKey}");
                        Debug.WriteLine($"Successfully Authenticated Key: {clientPublicKey}");
                    }
                    else
                    {
                        Console.WriteLine($"Warning unable to authenticate key. The ring was not updated.");
                        Debug.WriteLine($"Warning unable to authenticate key. The ring was not updated.");
                    }
                }
            }
        }

        /*
        /// <summary>
        /// Helper Class for Giving Debug.Write Functionality to other Resources..
        /// </summary>
        public static class Helper
        {
            public static void Write(string message)
            {
                Debug.WriteLine(message);
            }

            public static void Write(params object[] list)
            {
                if (list.Length == 0)
                {
                    Debug.WriteLine();
                }
                else if (list.Length == 1)
                {
                    Debug.WriteLine(string.Format("{0}", list[0]));
                }
                else
                {
                    Debug.WriteLine(string.Format(list[0].ToString(), GetArgs
                     (list)));
                }

            }

            /// <summary>
            /// From Array Extracts 2nd..Last
            /// </summary>
            private static object[] GetArgs(object[] list)
            {
                int argCount = list.Length - 1;
                object[] args = new object[argCount];
                for (int i = 0; i < argCount; i++)
                {
                    args[i] = list[i + 1];
                }
                return args;
            }
        }
        */

    }
}
```