ภาคผนวก

ภาคผนวก ก

โค้ดสำหรับใช้ Service ของ AWIS ของ TrafficHistory

```
namespace UrlInfoWriteHistory
{
    public class UrlInfo
    {
        public static readonly string SERVICE_HOST = "awis.amazonaws.com";
        public static readonly string SIGNATURE_VERSION = "2";
        public static readonly string SIGNATURE_METHOD = "HmacSHA256";
        public static readonly string ACTION = "TrafficHistory";
        public static readonly string RESPONSE_GROUP = "History";
        private string accessKeyId;
        private string secretAccessKey;
        private string site;
        public UrlInfo(string accessKeyId, string secretAccessKey, string site)
        {
            this.accessKeyId = accessKeyId;
            this.secretAccessKey = secretAccessKey;
            this.site = site;
        }
        public class OrdinalComparer : IComparer
        {
            int IComparer.Compare(Object x, Object y)
            {
                return String.CompareOrdinal(x.ToString(), y.ToString());
            }
        }
        private string makeSignature(string data)
        {
            string str_to_sign = "GET\n" + SERVICE_HOST + "\n/\n" + data;
            System.Console.WriteLine("String To Sign: \n" + str_to_sign);
            Encoding ae = new UTF8Encoding();
            HMACSHA256 signature = new HMACSHA256(ae.GetBytes(this.secretAccessKey));
            string b64 = Convert.ToBase64String(signature.ComputeHash(ae.GetBytes
                    (str_to_sign.ToCharArray())));
    return b64;
}
private static string getTimestamp()
{
    return System.DateTime.UtcNow.ToString("yyyy-MM-ddTHH:mm:ss.fffZ",
            System.Globalization.CultureInfo.InvariantCulture);
}
private string composeUrl(string timestamp,string range,string start)
{
    SortedList query = new SortedList(new OrdinalComparer());
    query.Add("Action", ACTION);
    query.Add("AWSAccessKeyId", accessKeyId);
    query.Add("ResponseGroup", RESPONSE_GROUP);
    query.Add("Timestamp", timestamp);
    query.Add("SignatureVersion", SIGNATURE_VERSION);
    query.Add("SignatureMethod", SIGNATURE_METHOD);
    query.Add("Url", site);
    query.Add("Range", range);
    query.Add("Start", start);
    List<string> query_arr = new List<string>();
    foreach (DictionaryEntry de in query)
    {
        query_arr.Add(de.Key.ToString() + "=" + URLEncode(de.Value.ToString()));
    }
    return String.Join("&", query_arr.ToArray());
}
private static string URLEncode(string str)
{
    Regex rgx = new Regex(@"[^A-Za-z0-9\-_.~]");
    MatchCollection matches;
    string encoded = "";
```

```
            foreach (char c in str.ToCharArray())
            {
                matches = rgx.Matches(c.ToString());
                if (matches.Count > 0)
                {
                    int tmp = c;
                    encoded += "%" + String.Format("{0:x2}",
(uint)System.Convert.ToUInt32(tmp.ToString())).ToUpper();
                }
                else
                {
                    encoded += c;
                }
            }
            return encoded;
        }
public static String urlConcastAlexa(String url, string accessKeyId, string secretAccessKey,string
range,
                                                            string start)
        {
            string site = url;
            UrlInfo urlInfo = new UrlInfo(accessKeyId, secretAccessKey, site);
            string timestamp = getTimestamp();
            string query = urlInfo.composeUrl(timestamp,range,start);
            string signature = urlInfo.makeSignature(query);
            url = "http://" + SERVICE_HOST + "/?" + query + "&Signature=" +
URLEncode(signature);
            return url;
        }
    }
}
```