WebServices Ojos del Cielo:

Con el webservice se puede pedir información para su sistema, por ej, pedir la posición de un móvil, u obtener los móviles cercanos a una coordenada. También permite que aportes información a nuestro sistema para visualizarla gráficamente en el mapa de monitoreo de móviles, por ej, cambiar el estado de un móvil de libre a urgencia.
Todas las llamadas tienen implementado un esquema de seguridad que utiliza clave asimétrica.
Para cada llamada al webservice hay que pasar cliente, fechaYHora y la clave generada como parametros como mínimo.

El siguiente codigo C# genera una firma válida. Esa firma es valida para el privateKey, cliente, y fechaYHora generados. Al hacer la invocación al webservice hay que usar los últimos dos y la firma.

```
/// <summary>
/// Genera una firma para usar en los metodos web.
/// </summary>
/// <param name="formattedTime">En C# es la hora actual en este formato:
DateTime.Now.ToString("yyyyMMdd HH:mm:ss:fff")</param> /// <param name="ipPublica">Ip publica
WAN (no de la red local interna).
Se puede consultar accediendo a
http://www.lineasodc.com.ar/whatismyip.aspx usando el valor que muestre REMOTE_ADDR.</param>
/// /// <param name="privateKey">Clave que les fue asignada por Ojos del Cielo.</param> ///
<returns></returns> protected static string Firmar(string ipPublica, string formattedTime,
string privateKey) {
     var textToSign = ipPublica + "," + formattedTime;
     var firma = Sign(textToSign, privateKey);
     return firma;
}

/// <summary>
/// Genera una firma en base a un texto y una clave.
/// </summary>
/// <param name="text">Texto</param>
/// <param name="keyString">Clave</param> /// <returns></returns> public static string
Sign(string text, string keyString) {
     var encoding = new System.Text.ASCIIEncoding();

     // converting key to bytes will throw an exception, need to replace '-' and '_'
characters first.
     string usablePrivateKey = keyString.Replace("-", "+").Replace("_", "/");
     byte[] privateKeyBytes = Convert.FromBase64String(usablePrivateKey);

     var textBytes = ASCIIEncoding.ASCII.GetBytes(text);
     // compute the hash
     var algorithm = new
System.Security.Cryptography.HMACSHA1(privateKeyBytes);
     byte[] hash = algorithm.ComputeHash(textBytes);

     // convert the bytes to string and make url-safe by replacing '+'
and '/' characters
     string signature = Convert.ToBase64String(hash).Replace("+",
"-").Replace("/", "_");
     return signature;
}
```