VPAR SERVER

Integration Manual

Databases

www.neurallabs.net

VERSION CONTROL			
	Version	**Date**	**Change Description**
Document history	1.0	23/08/2013	First version (MA).
	1.1	01/01/2014	XML format changes (MA)
	1.2	01/10/2014	XML format changes (IVAN)
	1.3	23/10/2014	Default queries incorporation
	1.4	01/06/2015	New types of triggers
	1.5	24/02/2017	General review (MAR)
		10/3/2017	General review (LL)

Introduction

VPAR SERVER is an IP camera based plate recognition system. Its main feature is the detection of license plates from multiple video sources. The system hosts the results in an open database (SQL Server).

VPAR SERVER supports multiple IP camera protocols:

- RTSP
- GigeVision
- Motion JPG
- Directshow
- JPG
- Other proprietary protocols

Also allows to connect to:

- Videos AVI
- Images folder
- Video channels from Monitoring Stations like:
    - Milestone
    - IndigoVision
    - Others

For more information about use and functionalities of VPAR SERVER, please refer to user manual.

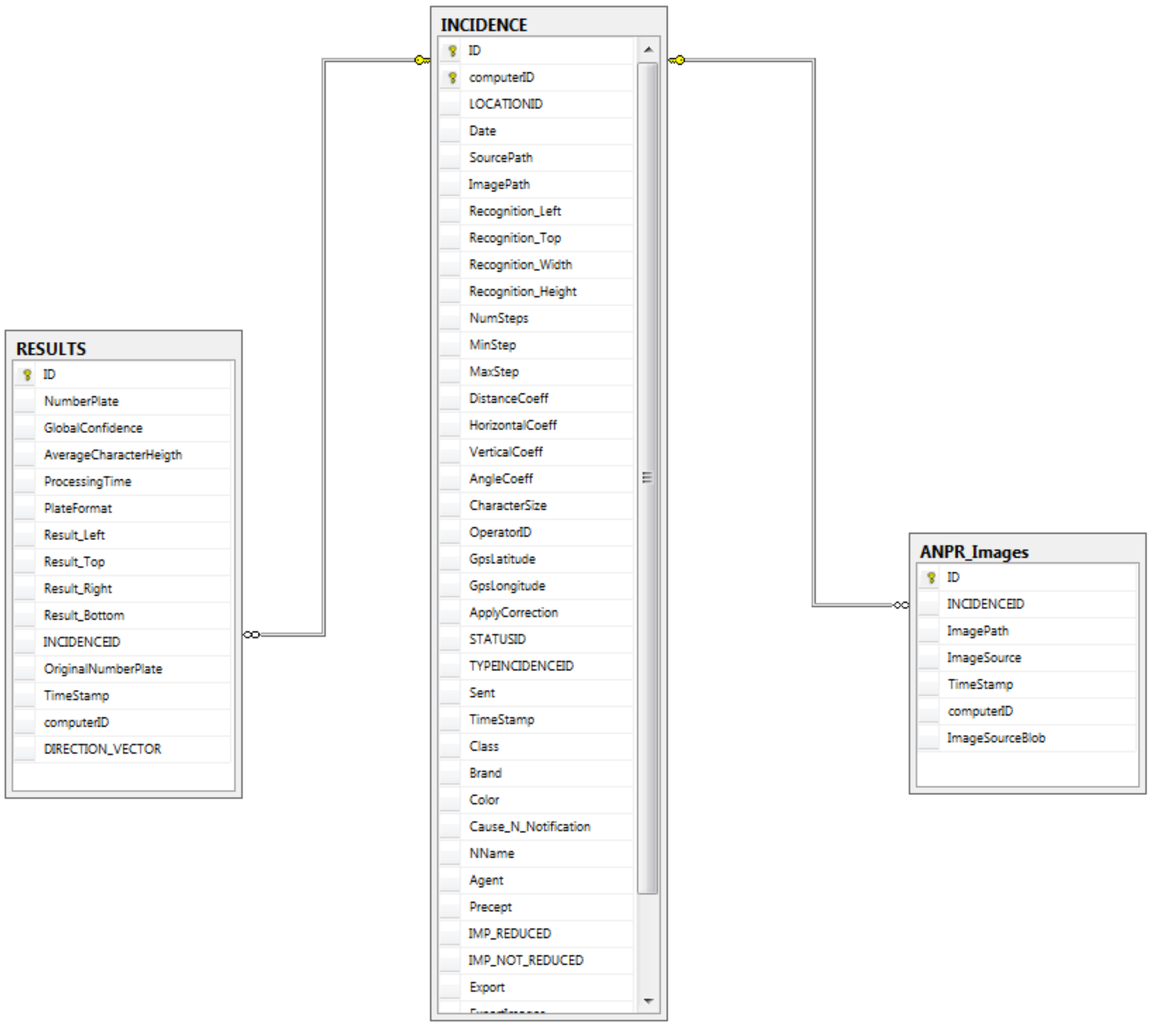
This document is intended to assist the user on how to integrate with the VPAR SERVER database for:

- Retrieve data from plate recognition system.
- Manage plate blacklists.

Queries for retrieve plate recognition data

VPAR SERVER stores the results of plate recognition in a database running on Microsoft SQL Server manager. This is an open database, so the integrator can access to it and extract data (in real time or through historical queries).

The following diagram shows the main tables involved in the data storage, and their relations:



INCIDENCE table:

This table stores the *general* data associated to a recognized plate. Each table entry (incidence) corresponds to a valid plate reading. The most significant columns of each *incidence* are:

- computerID: Unique identification of the computer where the plate has been detected.
- ID: Incidence identifier for a given computer ID (PK).
- Date: Date and time of the incidence.
- ImagePath: Path of the image file in which the plate was detected.
- GpsLatitude: Latitude coordinate detection (if the option is activated).
- GpsLongitude: Longitude coordinate detection (if the option is activated).
- camera_id: Unique identification of the camera from which the plate has been detected (CAMERAS master table).

RESULTS table:

This table stores the *particular* data associated to a recognized plate. Each entry corresponds to a valid detection, and has a related registry entry in the INCIDENCE table. The most significant columns are:

- computerID: Foreign key in relation to INCIDENCE table.
- INCIDENCEID: Foreign key in relation to INCIDENCE table.
- NumberPlate: Recognized plate.
- GlobalConfidence: Confidence rate.
- AverageCharacterHeigth: Average character height measured in pixels.
- ProcessingTime: LPR image processing time.
- PlateFormat: Plate format (country code).
- Result_Left: X left coordinate of the plate.
- Result_Top: Y top coordinate of the plate.
- Result_Right: X right coordinate of the plate.
- Result_Bottom: Y bottom coordinate of the plate.
- ID: Primary key of the table.
- TimeStamp: Exact record of the recognition time.
- DIRECTION_VECTOR: Flow direction of the plate. Possible values:
  - 0: Unknown direction or it has not moved.
  - -1: Moves away from the camera position.
  - 1: Approaching to the camera position.

ANPR_IMAGES table:

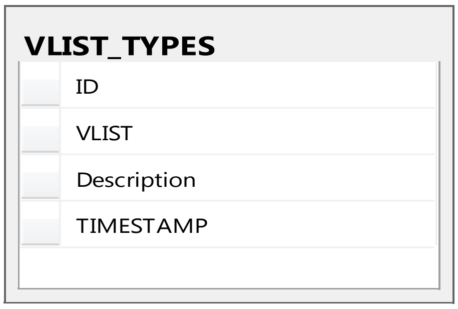
This table stores the original image in which the plate was detected (this feature will be running once the “Save Images in DB” option is checked on VPAR SERVER). This table is realted to the INCIDENCE “1 to 1” table. The most significant columns are:

- computerID: Foreign key in relation to INCIDENCE table.
- INCIDENCEID: Foreign key in relation to INCIDENCE table.
- ImageSourceBlob: Full image where the plate has been found (JPEG).

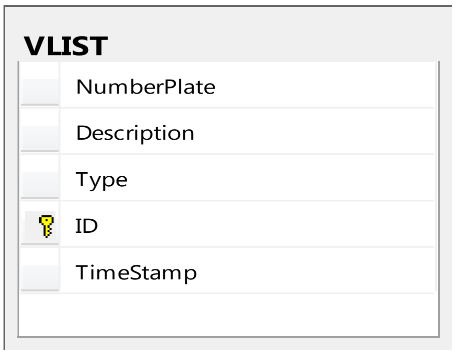
Blacklist

VPAR SERVER allows the creation of license plate lists which allows real time notifications about a certain find on this list.

The user can manage this lists from the following tables:



VLIST_TYPES
ID
VLIST
Description
TIMESTAMP



VLIST
NumberPlate
Description
Type
ID
TimeStamp

VLIST_TYPES table:

This table contains the possible list types. For example: List of vehicles under warrant, list of vehicles with expired insurance, list of vehicles without technical verification, etc. Then, each plate in the VLIST table will be related to a list type defined in this table. The most significant columns are:

- ID: Primary key of the table (AUTOMATIC).
- VLIST: Type of list identifier.
- Description: List description.
- TIMESTAMP: Exact creation time of the register (AUTOMATIC).

VLIST table

This table stores the plates that are in the blacklist. The columns of the table are defined as follow:

- NumberPlate: Plate number in blacklist.
- Description: Description associated to the plate in blacklist.

- Type: Foreign key related to VLIST TYPES (VLIST column of VLIST TYPES)
- ID: Primary key of the table (AUTOMATIC).
- TimeStamp: Exact time of the creation of the register (AUTOMATIC).

SQL Queries examples

The following section presents some SQL queries that the user can take as a basis for developing their own queries.

```sql
/*---------------------------*/
/* RECOGNIZED PLATES QUERIES */
/*---------------------------*/

-- Basic query
-----------------
SELECT * FROM incidence
JOIN results ON incidence.id = results.incidenceid AND incidence.computerID = results.computerID
LEFT JOIN anpr_images on incidence.id = anpr_images.incidenceid AND incidence.computerID =
anpr_images.computerID

--Example filters
--Dates
--WHERE incidence.TimeStamp BETWEEN '20140101 00:00:00' AND '20140102 23:59:59'

--Date and camera number
--WHERE incidence.TimeStamp BETWEEN '20140101 00:00:00' AND '20140102 23:59:59' AND incidence.camera_id  = 1

-- Date, camera numberand computer ID
--WHERE incidence.TimeStamp BETWEEN '20140101 00:00:00' AND '20140102 23:59:59' AND incidence.camera_id  = 1
AND incidence.computerid = 1

--Date and plate number
--WHERE incidence.TimeStamp BETWEEN '20140101 00:00:00' AND '20140102 23:59:59' AND results.numberplate =
'AAA111'

-- Date and plate number in black list
--WHERE incidence.TimeStamp BETWEEN '20140101 00:00:00' AND ' 23:59:59' AND EXISTS (SELECT * FROM vlist WHERE
vlist.NumberPlate  = results.NumberPlate)


/*--------------------*/
/* BLACK LIST QUERIES */
/*--------------------*/

-- Basic query
-----------------
SELECT * FROM vlist
JOIN vlist_types ON vlist.Type = vlist_types.vlist

--Example filters
--Plate number
--WHERE vlist.NumberPlate = 'AAA111'

--Plate number and list type
--WHERE vlist.NumberPlate = 'AAA111' AND vlist.type = 10

--List type insertion query (vlist_type)
--------------------------------------------------------
INSERT vlist_types (vlist, description)
VALUES (10, 'Black List')

--Insertion of plate number in black list (vlist)
---------------------------------------------------------------------
INSERT vlist (numberplate, description, type)
VALUES ('AAA111', '1001 SEARCH CODE', 10)

--Delete plate from 'Black List' (vlist)
-----------------------------------------------------------------------
DELETE FROM vlist
WHERE numberplate = 'AAA111' AND type = 10
```

MESSAGE SENDING

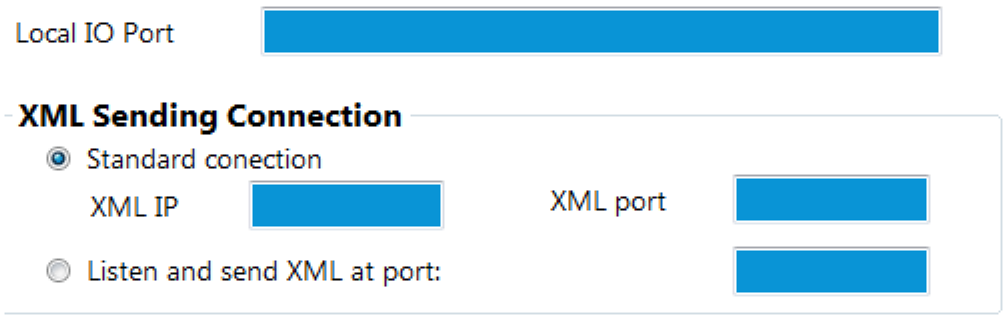
Send configuration

To configure the XML sending, the following parameters must be configured:

- IP of a server listening to a socket in **XML IP**
- Port on this server in **XML PORT**

In case we want VPAR SERVER to listen to a port and to send the XML data through that port, we will activate the option “Listen and send XML at Port”. **Attention, this mode disable the normal XML sending**

We can configure the connection closing after sending with the "Close connection after send" option.


Local IO Port

XML Sending Connection

[✓ Standard conection]

XML IP

XML port

Listen and send XML at port:

XML format

XML format is:

NEURAL + XML_SIZE + XML

An example of this message would be (consider that we are using random values):

NEURAL ⇓ Fixed tag.

234231 ⇓ Size (in bytes) of the XML message

<?xml version=""1.0"" encoding=""utf-8""?>

```
<infoplate>
  <DateHour>2014-09-27 11:15:40.000</DateHour>
  <Plate>B3809WH</Plate>
  <Confidence>99.99</Confidence>
  <Country>SPAIN</Country>
  <CountryID>101</CountryID>
  <ProcTime>120</ProcTime>
  <CamID>2</CamID>
  <ImageLeft>23</ImageLeft>
  <ImageTop>45</ImageTop>
  <ImageWidth>148</ImageWidth>
  <ImageHeight>71</ImageHeight>
 <Path>c:\\tmp\\imgs\\874A8E0A-C8A6-4D6A-9360-
ADB09A3FCD69\\20140927\\1115409401644-000001-18553032-GCA7766.jpg
</Path>
   <DirectionVector>-1</DirectionVector>
  <imgSize>0</imgSize>
  <img></img>
  <ComputerID>1</ComputerID>
  <Feedback>OK</Feedback>
 <ID_Event></ID_Event>
 <IncidenceID></IncidenceID>
 <nFrame></nFrame>
 <aviFileName></aviFileName>
 <Code></Code>
 <Speed></Speed>
<GPSLat></GPSLat><GPSLon></GPSLon>
<List></List>
<Evidences>
        <Evidence>
                <IdCamEv>2</IdCamEv>
                 <imgEVSize></imgEVSize>
                 <imgEV></imgEV>
        <Evidence>
<Evidences>

</infoplate>
```

(Please, note that in case there is no image, 'imgSize' will be 0 and the 'img' tag will be an empty field. If we want to send images, ‘Send image’ checkbox on ‘Configuration’ tab must be checked)

Where:

Date Hour:	Date and time of the recognition.
Plate:	Recognized plate.
Confidence:	Recognition confidence.
Country:	Name of the recognized country according to the file of identifiers and descriptions "countrycodes.txt".
CountryID:	Recognized Country ID according to “CountryCodes.txt”.
ProcTime:	Image processing time.
CharHeight:	Average height of the characters in pixels.
CamID:	Identifier of the camera from which the license plate has been read.
CamName:	Name of the camera which has performed the reading.
ImageLeft:	X left coordinate of the plate.
ImageTop:	Y top coordinate of the plate.
ImageWidth:	Plate width (in pixels).
ImageHeight:	Plate height (in pixels).
Path:	Local folder where the image will be stored (in the VPAR SERVER computer).
DirectionVector:	Direction of the vehicle: 1 Approaching/ -1 Moving away.
imgSize:	Image size (in bytes). 0 not included.
img:	Image (in JPEG).
ComputerID:	Computer ID that has performed the recognition.
Feedback:	Reason of NO reading, in case of using a trigger (plate not detected, plate has not passed filters, etc).
ID_Event:	Event ID of requested petition.
IncidenceID:	Consecutive recognition identifier. Unique along with ComputerID.
nFrame:	When we process Video AVI, frame number where the recognition was made.
aviFileName:	When we process Video AVI, file used to recognize.
Speed:	Instant Speed of the Vehicle if the NL Speed option is activated.
GPSLat:	Detection GPS latitude.
GPSlon:	*Detection GPS Length.*
List:	ID list or -1 if the plate is not in list
Evidences:	Set of evidence images detected.

- Evidence: Evidance image information.

    - IdCamEv: Evidence camera identifier.
    - *imgEVSize: Number of bytes of the associated environment camera image, in bytes (jpg) / 0 if not included.*
    - *imgEV:* Image from the environment camera in jpg, if there is any.

TRIGGER FORMAT (SENT TO VPARSERVER)

For the trigger activation in VPAR SERVER, software needs to receive an XML message in the LOCAL PETITION PORT. This message must follow this format:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Alarm>
<Source>
        <IdSource>”IDSOURCE”</IdSource>
</Source>
<AlarmDetail>
<AlarmType>”ALARMTYPE”</AlarmType>

<ExtraFields>
  <EF Name="CameraMask">”CAMERAMASK”</EF>
    </ExtraFields>
    </AlarmDetail>
    </Alarm>
```

Where:

CameraMask:	It is a byte mask in which each position indicates the camera to shoot. 2 exp (num-1). Eg Camera 1 = 1, Camera 2 = 10, Camera 3: 100, etc.
IDSOURCE:	Alphanumeric value that identifies the trigger. This value will be returned in the response in the “Id_Event" field. (This tag is optional).
ALARMTYPE:	Type of alarm:

- **0:** Standard Trigger: VPAR SERVER will capture X seconds or X images (depending on VPAR SERVER configuration) and will return the reading result (first valid recognition or the NO reading)
- **10:** Initial capture Trigger: VPAR Server will begin capturing indefinitely until a valid result is found, or until a 20 type event is received.
- **20**: Stop capture Trigger: VPAR SERVER will stop the capturing initiated by 10 or 30 triggers.
- **30:** Multiple result Init capture Trigger: VPAR SERVER will begin capturing indefinitely, returning all valid results until receiving a 20 type trigger.
- **40:** Multiple result standard trigger: will capture X seconds or X images (depending on VPAR SERVER configuration) and will return all reading results (all recognitions, or NO reading results).

Webservices

Inside the installation package, a web package is incorporated, that will deploy a webservice for the data query.

Webservices instalation

For the installation of webservices follow these steps:

1. Install the ticket creator in the Windows event log. For this, execute:

   C: \ WINDOWS \ Microsoft.NET \ Framework \ v4.0.30319> InstallUtil.exe INSTALLATION_PUT \ Webservices \ Log \ EventLogSourceInstaller.dll

2. Copy the folder: INPUT_TRANSPORT \ Webservice \ WebserviceDataManager to C: \ inetpub \ wwwroot.

3. Edit the Web.config file and edit the following parameters:

   a. Database connection:

   <add name="ANPRConnectionString" connectionString="Data Source= localhost\VPARSQLSERVER;Database=ANPR;Persist Security Info=True;User ID=sa;Password=VparAdmin01#" providerName="System.Data.SqlClient" />

   b. Country file:

   ```
   <setting name="CountryFolder" serializeAs="String">
       <value>C:\inetpub\wwwroot\WebserviceDataManager</value>
     </setting>
   ```

4. Open the IIS administrator “inetmgr.exe”
5. Expand the tree and we will see the folder “WebserviceDataManager” in the “Default web site” node.
6. Right click on "WebserviceDataManager" and then click "convert" to application.

Webservice use

Once installed, the web server will provide access to the web service. By default the url will be:

http://localhost/WebserviceDataManager/VparDataConsultor.asmx

and its definition for its implementation in different programming systems:

http://localhost/WebserviceDataManager/VparDataConsultor.asmx?wsdl

GetIncidenceInfoPlate

This function will return the information of a registration according to the entered data.

Entry parameters:

From	Start search date
To	Final search date
Inblacklist	Boolean that we can check if we want to search only for items that are in the list.
Plate	Plate to be searched. We can also search for fragments of it.

Parámetros de retorno:

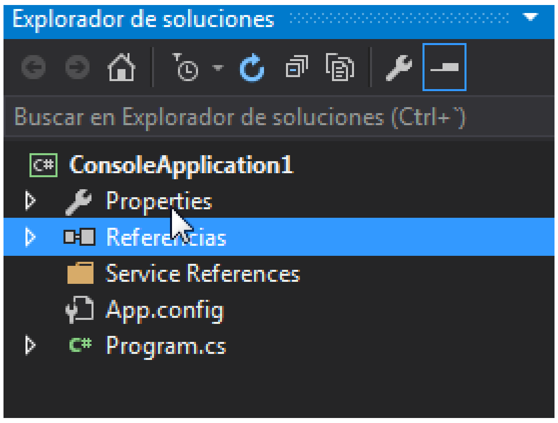
Id	Incidence identifier
DateTime	Incidence date
PlateNumber	Plate
CamName	Camera name
CamId	Camera identifier
ComputerID	VPAR SERVER instance id
Location	Camera location id
CountryName	Detected country name
LocationName	Camera location
Confidence	Reading Reliability
Speed	Speed in km/h
SpeedConfidence	Speed Reading reliability
DirectionVector	Vehicle direction
PlateImage	Image of the capture in base64

lengthImage	Image size
Observation	Observations of the incidence
ResultLeft	Left coordinate of the license plate in the image
ResultRight	Right coordinate of the license plate in the image
ResultTop	Top coordinate of the license plate in the image
ResultBottom	Bottom coordinate of the license plate in the Image
CountryId	Dectected country id
IsInBlackList	Indicator if exists in any list
typeId	List id
typelist	List name
descriptionbl	Description of the license plate within the list
commentsbl	Comments of the license plate within the list

Integration of web service with Visual Studio

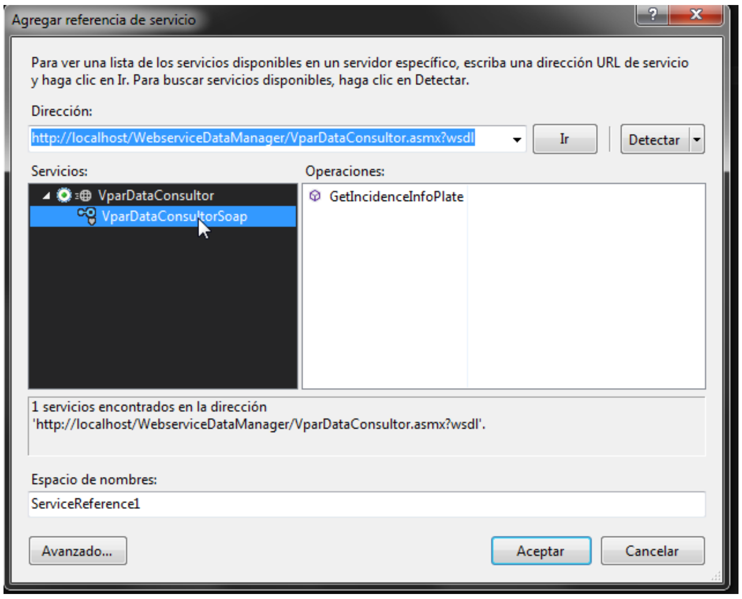
Example to deploy the service in Visual Studio:

- Verify the service is active by entering the address in the browser. For this example:
- http://localhost/WebserviceDataManager/VparDataConsultor.asmx?wsdl
- Create a new console application type project.
- In the Solution Explorer, right-click on "References" and select "Add Service Reference":



- Enter the service address in "Address".

- Select "Go" to connect to the service and then select "VparDataConsultorSoap":



Agregar referencia de servicio

Para ver una lista de los servicios disponibles en un servidor específico, escriba una dirección URL de servicio y haga clic en Ir. Para buscar servicios disponibles, haga clic en Detectar.

Dirección: http://localhost/WebserviceDataManager/VparDataConsultor.asmx?wsdl

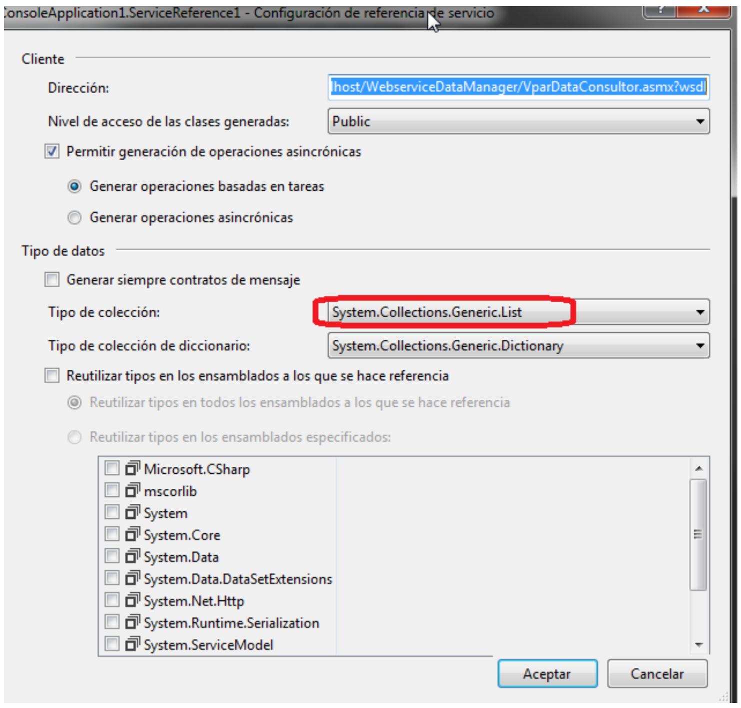
Servicios:
- VparDataConsultor
  - VparDataConsultorSoap

Operaciones:
- GetIncidenceInfoPlate

1 servicios encontrados en la dirección 'http://localhost/WebserviceDataManager/VparDataConsultor.asmx?wsdl'.

Espacio de nombres: ServiceReference1

- Review “Avdvanced…”:



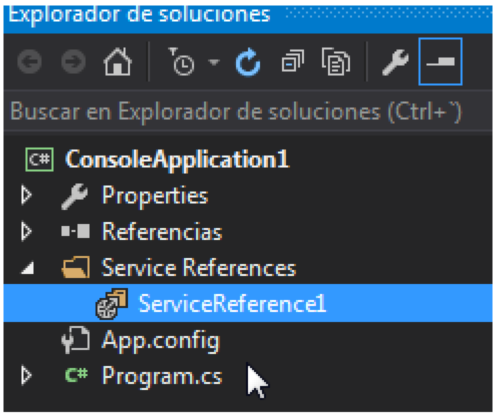
ConsoleApplication1.ServiceReference1 - Configuración de referencia de servicio

Cliente
Dirección: host/WebserviceDataManager/VparDataConsultor.asmx?wsdl
Nivel de acceso de las clases generadas: Public
[✓] Permitir generación de operaciones asincrónicas
[●] Generar operaciones basadas en tareas
[ ] Generar operaciones asincrónicas

Tipo de datos
[ ] Generar siempre contratos de mensaje
Tipo de colección: System.Collections.Generic.List
Tipo de colección de diccionario: System.Collections.Generic.Dictionary
[ ] Reutilizar tipos en los ensamblados a los que se hace referencia
[●] Reutilizar tipos en todos los ensamblados a los que se hace referencia
[ ] Reutilizar tipos en los ensamblados especificados:
- [ ] Microsoft.CSharp
- [ ] mscorlib
- [ ] System
- [ ] System.Core
- [ ] System.Data
- [ ] System.Data.DataSetExtensions
- [ ] System.Net.Http
- [ ] System.Runtime.Serialization
- [ ] System.ServiceModel

- Finally Accept to create the service referente.

- A new service reference will be displayed in the project browser:



- In App.config check that the link has sufficient capacity to host large messages and increase if necessary:

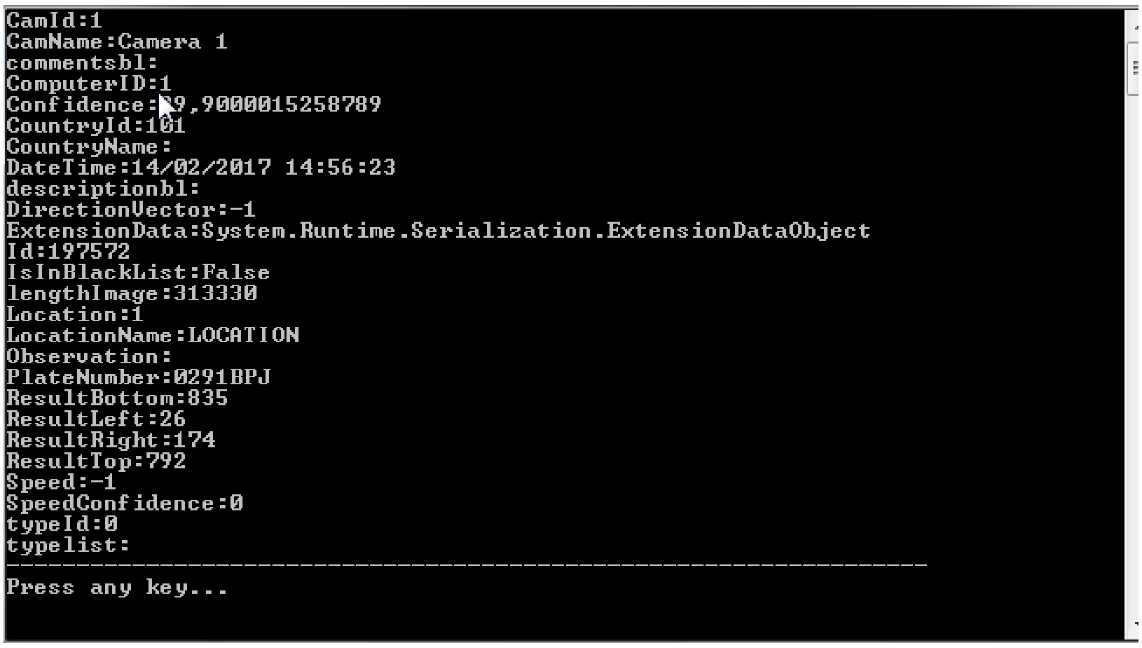
```
<binding name="VparDataConsultorSoap" maxBufferSize="2147483647"
maxReceivedMessageSize="2147483647" maxBufferPoolSize="524288"/>
```

- Edit Program.cs and enter the following code:

```
ServiceReference1.VparDataConsultorSoapClient srv1 = new
ServiceReference1.VparDataConsultorSoapClient();
            List<ServiceReference1.Incidence> result = new
List<ServiceReference1.Incidence>();

            ServiceReference1.GetIncidenceInfoPlateRequest request = new
ServiceReference1.GetIncidenceInfoPlateRequest();
            request.Body = new ServiceReference1.GetIncidenceInfoPlateRequestBody();

            List<ServiceReference1.Incidence> response = new
List<ServiceReference1.Incidence>();

            response = srv1.GetIncidenceInfoPlate(new DateTime(2017, 2, 14, 14, 56, 23), new
DateTime(2017, 2, 14, 14, 56, 24), false, "1234ABC");

            foreach(var register in response)
            {
                Console.WriteLine("CamId:" + register.CamId);
                Console.WriteLine("CamName:" + register.CamName);
                Console.WriteLine("commentsbl:" + register.commentsbl);
                Console.WriteLine("ComputerID:" + register.ComputerID);
                Console.WriteLine("Confidence:" + register.Confidence);
                Console.WriteLine("CountryId:" + register.CountryId);
                Console.WriteLine("CountryName:" + register.CountryName);
                Console.WriteLine("DateTime:" + register.DateTime);
                Console.WriteLine("descriptionbl:" + register.descriptionbl);
                Console.WriteLine("DirectionVector:" + register.DirectionVector);
                Console.WriteLine("ExtensionData:" + register.ExtensionData);
                Console.WriteLine("Id:" + register.Id);
                Console.WriteLine("IsInBlackList:" + register.IsInBlackList);
                Console.WriteLine("lengthImage:" + register.lengthImage);
                Console.WriteLine("Location:" + register.Location);
                Console.WriteLine("LocationName:" + register.LocationName);
                Console.WriteLine("Observation:" + register.Observation);
                //Console.WriteLine(":" + register.PlateImage);
                Console.WriteLine("PlateNumber:" + register.PlateNumber);
                Console.WriteLine("ResultBottom:" + register.ResultBottom);
                Console.WriteLine("ResultLeft:" + register.ResultLeft);
                Console.WriteLine("ResultRight:" + register.ResultRight);
                Console.WriteLine("ResultTop:" + register.ResultTop);
```

```
        Console.WriteLine("Speed:" + register.Speed);
        Console.WriteLine("SpeedConfidence:" + register.SpeedConfidence);
        Console.WriteLine("typeId:" + register.typeId);
        Console.WriteLine("typelist:" + register.typelist);

        Console.WriteLine("-----------------------------------------------");
    }

    Console.WriteLine("Press any key...");
    Console.ReadLine();
```

- Change date and plate values to correct values.
- When running, values similar to the following should appear:



```
CamId:1
CamName:Camera 1
commentsbl:
ComputerID:1
Confidence:99,9000015258789
CountryId:101
CountryName:
DateTime:14/02/2017 14:56:23
descriptionbl:
DirectionVector:-1
ExtensionData:System.Runtime.Serialization.ExtensionDataObject
Id:197572
IsInBlackList:False
lengthImage:313330
Location:1
LocationName:LOCATION
Observation:
PlateNumber:0291BPJ
ResultBottom:835
ResultLeft:26
ResultRight:174
ResultTop:792
Speed:-1
SpeedConfidence:0
typeId:0
typelist:
-----------------------------------------------------------------
Press any key...
```