

# YourCarMotorInsurance Webservice

Codeweavers Ltd provides a number of vehicle finance and insurance services for use on the web and the desktop. YourCarMotorInsurance is part of our YourCar suite of products and the Webservice, in conjunction with LloydLatchford Insurance, provides instant, customised, motor insurance quotes to any website using webservices.

## DOCUMENTATION version 1.3

### Table of Contents

Table of Contents	1
Introduction	2
The Webservice	2
C# Example	2
Java	2
PHP	2
ASP Classic	2
Coldfusion	2
SOAP Input	3
SOAP Example	3
Customer Data	3
Vehicle Data	4
SOAP Actions	4
SOAP Response	5
CalculateAndRender	5
Transforming the Output	6
XSL Parameters	6
The JavaScript Object	7
Overriding Default Values	7
Listing Calculations	7
CSS Hooks for Listing Calculations	7
Insurance Group	7
Insurance Group Container	8
Capid Based Listing	8
Full Insurance Calculation	8
Custom Form Rendering	8
Fulfilment Page	9
Additional Support	10
Codeweavers Ltd	10
Third Party Resources	10

## Introduction

In addition to access to the Webservice we can also supply you with code and XML examples, along with transformation scripts and documentation to use them.

The package basically consists of:

- the webservice that provides 3 types of motor insurance calculation
- a javascript popup that aggregates a list of insurance groups and performs asynchronous requests to the web service proxy
- a proxy page that calls the webservice and transforms the returned XML for both the list calculation and single vehicle calculation

## The Webservice

URL: <https://secure.codeweavers.net/miwebservices/InsuranceCalculation.asmx>

WSDL: <https://secure.codeweavers.net/miwebservices/InsuranceCalculation.asmx?WSDL>

Written in C# .NET, the webservice uses SOAP and a full SOAP envelope architecture to receive and return data. The data input is defined in the WSDL document which can be used to validate the input using schema validation, or can be used to generate objects in .Net, Java and PHP. Other technologies such as ColdFusion can pass in Struct in the format of the SOAP input, or Classic ASP or similar technologies can send a POST request with the SOAP envelope encoded.

### C# Example

The provided C# example uses the Visual Studio.NET web service connection wizard. This connects to the webservice, finds the WSDL, and generates the required files. These files provide a webservice object which has method calls for each of the webservice methods; `Calculate`, `CalculateAndRender`, `CalculateList` and `CalculateListFromGroups`. It also provides the required data structures needed to populate the webservice SOAP request; `InsuranceCalculationData` and `InsuranceCalculationVehicle`.

### Java

For java most professional level integrated development environments (like Netbeans or Eclipse) have discovery services that will create skeleton classes from the WSDL

### PHP

WSDL services can generate the stub classes as in the C# example but lighter libraries such as the open source nusoap library can call the webservice using a standard PHP associative array.

### ASP Classic

Rather than having a built in SOAP library ASP utilises a standard POST request and sets the contenttype and SOAPAction request headers accordingly. The post body is then populated with the SOAP envelope as XML, following the structure as defined on the operation page which can be viewed by going to the webservice address in a web browser and then selecting the operation.

### Coldfusion

This example is in development but follows a similar paradigm to the PHP example in that you create a struct based on the SOAP input XML and use the built in WSDL based webservice consumer.

## SOAP Input

The input for the latest version of the webservice can be viewed on the webservice operations pages. The basic structure is the same for the authentication data (username, password) and the customer data for all webservice methods. The vehicle data takes different arguments for each method as the `calculate` takes a single vehicle, the `calculatelist` takes an array of vehicles and the `calculatelistfromgroups` takes an array of integers.

## SOAP Example

The example below relates to the `Calculate` method but can be easily translated by referring to the webservice pages.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Calculate xmlns="http://www.lloydlatchford.com/webservices/">
      <Username>string</Username>
      <Password>string</Password>
      <CustomerData>
        <HasAccident>boolean</HasAccident>
        <HasConviction>boolean</HasConviction>
        <IsGaraged>boolean</IsGaraged>
        <DateOfBirth>string</DateOfBirth>
        <Mileage>int</Mileage>
        <Gender>boolean</Gender>
        <RestrictionsId>int</RestrictionsId>
        <PostCode>string</PostCode>
        <PromoCode>string</PromoCode>
        <Title>string</Title>
        <Forename>string</Forename>
        <Surname>string</Surname>
        <Email>string</Email>
        <Telephone>string</Telephone>
        <Building>string</Building>
        <BuildingName>string</BuildingName>
        <NoClaimsDiscount>int</NoClaimsDiscount>
      </CustomerData>
      <VehicleData>
        <CapId>int</CapId>
        <OptionalInsuranceGroup>int</OptionalInsuranceGroup>
        <VehicleAge>int</VehicleAge>
        <VehicleYear>int</VehicleYear>
        <OptionalVehicleReg>string</OptionalVehicleReg>
        <OptionalVehicleMileage>int</OptionalVehicleMileage>
      </VehicleData>
    </Calculate>
  </soap:Body>
</soap:Envelope>
```

## Customer Data

The username, password and site id will be supplied to you separately, and the customer data is collected using the popup form, or from your own form. In either case it should be stored in a cookie and sent via POST to the proxy page. The cookie ensures persistence throughout the site, and also between sessions – so a user can come back to your site and instantly receive their custom insurance quotation based on their previous details. Any provided form should allow editing of the credentials that then updates the cookie values. This functionality is provided by the javascript popup.

To allow for integration into system with registration/members etc an additional function call is made when the cookie information is ready to be saved. This returns true by default but can be used to call other functions such as registering a user. This can then return false and prevent the saving if, for instance, the registration failed.

The `CustomerData` structure corresponds directly to the user input form. Each value and type is simply populated either in a language structure or directly in the SOAP XML envelope.

The `PromoCode` feature allows you to send an optional code that provides either a percentage discount or uplift as arranged with LloydLatchford. There is also scope to pass this into the customer information directly from the user's input if your site is to offer varied discounts based on these codes.

### Vehicle Data

This structure is simply the `capid` and the optional insurance group although the cap dataset is complete there may be occasions when the `capid` fails, so the optional insurance group can be used to obtain an approximate quotation. IF the insurance group of a vehicle is readily available then it should be included in the input – however, only `capid` is required. The `capid` referred to here is the 5 digit integer, and not the longer string of alpha numeric characters, known as the cap code.

### SOAP Actions

The method signatures for the webservice actions are:

- Calculate
  - Username - string
  - Password - string
  - CustomerData - InsuranceCalculationData
  - VehicleData - InsuranceCalculationVehicle
- CalculateAndRender
  - Username - string
  - Password - string
  - IsNew - boolean
  - VehicleYear - integer
  - CustomerData - InsuranceCalculationData
  - VehicleData - InsuranceCalculationVehicle
- CalculateList
  - Username - string
  - Password - string
  - CustomerData - InsuranceCalculationData
  - VehicleDataArray - Array of InsuranceCalculationVehicle
- CalculateListFromGroups
  - Username - string
  - Password - string
  - CustomerData - InsuranceCalculationData
  - InsuranceGroups - Array of integers

## SOAP Response

The response text returned from the webservice as the `<soap:Body>` in the response envelope is a simple XML document. The format is the same for all the calculations. The only difference is the vehicle element, which returns either the `Capid` or the `InsuranceGroup` depending on what element was used to make the calculation.

This is an example of the returned XML from a single vehicle calculation. Note there is only one child `Calculation` element of the root `Calculations` element. For a listed calculation the returned XML will contain multiple `Calculation` elements.

```
<Calculations xmlns="">
  <Site>1</Site>
  <Calculation>
    <Vehicle>
      <Capid>33155</Capid>
      <Regyear>2006</Regyear>
    </Vehicle>
    <Companies>
      <Company id="3">
        <CampaignId>17139</CampaignId>
        <Name><![CDATA[Allianz Cornhill]]></Name>
        <LogoUrl>
          http://secure.codeweavers.net/motorinsurance/images/logos/cornhill.gif
        </LogoUrl>
        <MonthlyPremium>67.044654000</MonthlyPremium>
        <AnnualPremium>804.48</AnnualPremium>
        <RestrictionApplies>False</RestrictionApplies>
        <PremiumUnavailable>False</PremiumUnavailable>
      </Company>
      <Company id="4" bestquote="true">
        <CampaignId>17140</CampaignId>
        <Name><![CDATA[Norwich Union]]></Name>
        <LogoUrl>
          http://secure.codeweavers.net/motorinsurance/images/logos/norwichunion.gif
        </LogoUrl>
        <MonthlyPremium>51.58420517800000</MonthlyPremium>
        <AnnualPremium>618.96</AnnualPremium>
        <RestrictionApplies>False</RestrictionApplies>
        <PremiumUnavailable>False</PremiumUnavailable>
      </Company>
      <Company id="1">
        <CampaignId>17141</CampaignId>
        <Name><![CDATA[Zurich]]></Name>
        <LogoUrl>
          http://secure.codeweavers.net/motorinsurance/images/logos/zurich.gif
        </LogoUrl>
        <MonthlyPremium>61.17625216000000</MonthlyPremium>
        <AnnualPremium>734.16</AnnualPremium>
        <RestrictionApplies>False</RestrictionApplies>
        <PremiumUnavailable>False</PremiumUnavailable>
      </Company>
    </Companies>
  </Calculation>
</Calculations>
```

**Note:** Company attribute id is for internal use, this is un-required for your use.

For an insurance group based calculator the following is returned as the child of the `Vehicle` element

```
<Vehicle>
  <InsuranceGroup>18</InsuranceGroup>
</Vehicle>
```

## CalculateAndRender

If you call the `CalculateAndRender` method from the webservice the returned value is pre-rendered HTML using a default XSL stylesheet for those users who wish not to transform the output themselves.

## Transforming the Output

For a listing calculation initiated by the provided javascript library, the output is transformed by the proxy class which outputs pure javascript. This updates the content of the page based on the CSS class of an element (see section on the javascript)

The output from a single vehicle calculation (which should show more detail) is transformed by XSL. This transformation will need to be implemented by your proxy class, however, we have supplied an XSL stylesheet that will take the output and transform it into a HTML table.

In addition to the HTML, each company's insurance quotation contains a HTML form, with a number of hidden variables. These variables are used to POST the required information from the cookie values, and the calculation result, to the fulfilment website. The variables needed are therefore passed into the XSL stylesheet as parameters.

### XSL Parameters

The following XSL parameters need to be passed into the stylesheet so that these can be added to the form which eventually sends a POST request to the fulfilment pages at Lloydlatchford.

- accessPointId – this will be supplied and allows us to identify you
- capId – the capid of the vehicle
- claim – the accidents Boolean from the cookie
- conviction – convictions Boolean from the cookie
- coverytype – the id of the coverytype selected from the drop down
- dob – the user's date of birth taken from the cookie
- email – users email address from the cookie
- forename – from the cookie
- garage – whether the vehicle is garaged, from the cookie
- gender – in the cookie
- isNew – tells the fulfilment site whether to prompt for registration
- number
- mileage – cookied
- postcode – from the cookie
- siteId – the site id that determines the branding if the fulfilment pages
- surname – from the cookie
- title – the user's title from the cookie
- tracker – from the cookie
- vehicleAge – the registration year of the vehicle, older vehicles are generally cheaper to insure so this is important
- vehicleMileage – the current mileage of the vehicle
- vehicleReg – the registration number of the vehicle
- noClaimsDiscount – the number of years no claims discount the customer has ranging from 1 - 6

Optional fields that may need to be supplied, depending on your XSL transformation engine.

- vehiclePrice – the vehicle price if known (otherwise a box appears on the fulfilment page)
- telephone – helps to pre-populate the fulfilment form
- building – building number. This is automatically added to the fulfilment form if known
- buildingName – as above

## The JavaScript Object

We also provide a javascript object for your convenience. This is available by including the script line

```
<script src="http://js.codeweavers.net/insurance_quote.js"></script>
```

This script will firstly include the required libraries for the functionality to work. It will also create the object needed, and set the default values. Some of these values can be overridden. It then provides the needed functionality thereafter using "CSS hooks".

### Overriding Default Values

There are currently 5 values that can be overridden in the JavaScript object. Others may be exposed and if you have a sufficient level of technical knowledge in prototype based JavaScript objects you could utilise these if needed.

The main values are:

```
InsuranceQuoteDetails.prototype.PromoCode           = '';
InsuranceQuoteDetails.prototype.ListQuoteString     = '';
InsuranceQuoteDetails.prototype.FullQuoteEnterDetailsMsg = '';
InsuranceQuoteDetails.prototype.PopupInstructionsText = '';
```

- PromoCode – populates the promocode field in the webservice call
- ListQuoteString – This is the message displayed next to the quoted premium on the listings/search page. It uses a mixin to "inject" the insurance company name at the desired point. The mixin place holder is `#{company}` eg  
Your best monthly **CAR INSURANCE QUOTATION**, fixed for 3 years, on this vehicle from `#{company}` is
- FullQuoteEnterDetailsMsg – this text is used to display the message on a full quote page, if the user's details have not already been entered. The default for this opens the user information popup
- PopupInstructionsText – short instruction that is placed at the top of the popup box

### Listing Calculations

The listing calculations should be used on any search page, where there are multiple vehicles. It is designed to be lightweight and unobtrusive, and work with groups of vehicles in contrast to the full calculation which is designed for information and robustness.

### CSS Hooks for Listing Calculations

To enable easy integration into a number of systems the `InsuranceQuoteDetails` object uses CSS hooks (or CSS predefined CSS class names) to obtain and set it's data.

You simply apply a class name to a given element that contains the data required, or contains the area that the result needs to populate.

### Insurance Group

The insurance group of each vehicle should be stored in the value attribute of an input element. If this field is then assigned the CSS class of "insurancegroup" the javascript will detect and append that insurance group to the list of insurance groups it sends to the web service.

The resultant quotation will then be applied to the insurance group container for that given insurance group

## Insurance Group Container

This container will be populated with the finished quote string and is identified by the class name `insuranceGroup[group]Container` where `[group]` is the insurance group as numerics eg `insuranceGroup20Container`.

This classname can be attached to any type of container that supports the innerHTML javascript DOM attribute. We recommend a `div` but `span`, `td`, `p` or others can be used.

## Capid Based Listing

Although not currently implemented in the object, this will work similarly to the insurance group hooks. A `capid` class will define the input element containing the `capid` and a `capid[capid]Container` will be populated with the result. Implementation of this could easily be achieved by simply extending the current methods and using the CalculateList web service method.

## Full Insurance Calculation

When a single vehicle has been selected on your site you can retrieve a full calculation that will return all the insurance quotations from the companies available through the webservice. This gives the user the maximum amount of information.

The javascript object also provides an interface for the full calculation quotation the method, as seen below, takes the vehicles capid as an argument.

```
<script>
    var insuranceQuotes          = new InsuranceQuoteDetails();
    insuranceQuotes.FullQuotations([capid]);
</script>
```

## Custom Form Rendering

If you choose to render your own question form rather than using the supplied javascript form then there are several values that need to be supplied, these can be found below.

```
<select id="coverType" name="coverType">
  <option selected="selected" value="">Select Cover Types</option>
  <option value="1">Insured Only To Drive</option>
  <option value="3">Insured Plus 1 Named Driver Over 25</option>
  <option value="2">Insured Plus 2 Named Driver Over 25</option>
</select>
```

```
<select id="noClaimsDiscount" name="noClaimsDiscount">
  <option selected="selected" value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
  <option value="6">6+</option>
</select>
```

```
<select id="gender" name="gender">
  <option selected="selected" value="False">Male</option>
  <option value="True">Female</option>
</select>
```

## Fulfilment Page

When integrating the motor insurance into your website you will be required to redirect the customer to the fulfilment page once they have chosen a quote they wish to purchase. This link will need to redirect to the following URL.

<https://www.yourcarmotorinsurance.co.uk/home/quote.aspx>

This page will collect all of the required details and redisplay the quote to the customer allowing them to read through some additional policy details.

In order for this page to re-quote the following details are required.

- accessPointId – this will be supplied and allows us to identify you
- capId – the capid of the vehicle
- claim – the accidents Boolean from the cookie
- conviction – convictions Boolean from the cookie
- covertype – the id of the covertype selected from the drop down
- dob – the user's date of birth taken from the cookie
- email – users email address from the cookie
- forename – from the cookie
- garage – whether the vehicle is garaged, from the cookie
- gender – in the cookie
- isNew – tells the fulfilment site whether to prompt for registration
- number
- mileage – cookied
- postcode – from the cookie
- siteId – the site id that determines the branding if the fulfilment pages
- surname – from the cookie
- title – the user's title from the cookie
- tracker – from the cookie
- vehicleAge – the registration year of the vehicle, older vehicles are generally cheaper to insure so this is important
- vehicleMileage – the current mileage of the vehicle
- vehicleReg – the registration number of the vehicle
- noClaimsDiscount – the number of years no claims discount the customer has ranging from 1 - 6

Optional fields that may need to be supplied, depending on your XSL transformation engine.

- vehiclePrice – the vehicle price if known (otherwise a box appears on the fulfilment page)
- telephone – helps to pre-populate the fulfilment form
- building – building number. This is automatically added to the fulfilment form if known
- buildingName – as above

## Additional Support

### Codeweavers Ltd

There are a number of developers working on this project at Codeweavers, who can be contacted for additional information. We endeavour to answer questions as soon as possible but we are a busy development house so scheduling and other resource constraints may affect the speed of our response. The primary points of contact are:

Paul Shannon  
Web Development Team Leader  
[paul.shannon@codeweavers.net](mailto:paul.shannon@codeweavers.net)

Chris Knight  
Web Developer  
[christopher.knight@codeweavers.net](mailto:christopher.knight@codeweavers.net)

Roland Schaack  
Development Director  
[roland@codeweavers.net](mailto:roland@codeweavers.net)

### Third Party Resources

The javascript objects used in this project are based on Prototype.js. This facilitates a number of additional extensions to javascript to allow larger, data driven objects to be created. It also supplies a wrapper for the XMLHttpRequest object of modern browser which the Ajax objects to make calls back to the server without refreshing the page.

More information can be found at the API site:  
<http://prototypejs.org/>

or at the script.aculo.us site which is a further prototype based extension used here  
<http://script.aculo.us>

This project also uses prototype windows for the popup, which has an API web site:  
<http://prototype-window.xilinus.com/>

Additional resources on SOAP, XML, XSLT, XSL and the other technologies used should be readily available via a simply google search.