IDS Datashare – webAPI

Introduction

The following document will describe briefly the webAPI that IDS has made available for use by its customers to dynamically retrieve vehicle and associated data directly from the IDS servers.

Technology

The IDS webAPI is built using Microsoft ASP.Net Web API platform, which allows the creation of HTTP based web services that are accessible from a whole host of computer languages.

The API should be considered to be a RESTful API see:
http://en.wikipedia.org/wiki/Representational_state_transfer

The only HTTP verb accepted by the API will be the ***GET*** verb.

Initially data returned from API calls can either be returned in JSON/JSONP format or XML format, though over time the list of supported formats may also be extended. The Web API will perform content negotiation – please see http://www.asp.net/web-api/overview/formats-and-model-binding/content-negotiation for more details.

It is intended that the API is easily callable from client languages such as C#.Net, JavaScript, PHP – other languages are likely possible but the 3 mentioned are where most testing will be performed.

HTTP Messages – i.e. Calling the API

In its simplest form, calling the API is as simple as constructing a URL, however to keep the URL 'pure' you may choose to embed some URL fields within the HTTP request header as additional fields – specifically fields such as *CustomerId*, are better suited to be embedded within the request header as additional fields, than passed on the URL – though both will be supported by the API.

* *At the time of writing this document, it has been discovered that passing X-CustomerId as a custom HTTP request header is getting lost when sending the request by jQuery on a remote host – and therefore at this moment in time, the only reliable way to pass your customer id is as part of the URL -- Hopefully this issue will get addressed shortly* *

Success or failure of the API call is communicated back in the HTTP Response object:
Likely response codes will be:

200	***OK*** - Call was successful
401	***Unauthorised*** – Either CustomerId was not supplied or was incorrect - this may also be returned if an attempt was made to call API functionality to which you are not currently subscribed for.
404	***Not Found*** – Data could not be found i.e. if looking for a VRM and it wasn't found

There may be other codes that are implemented, these will be noted where they become possible return status codes – for a full list of potential status codes and their meanings refer to: http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

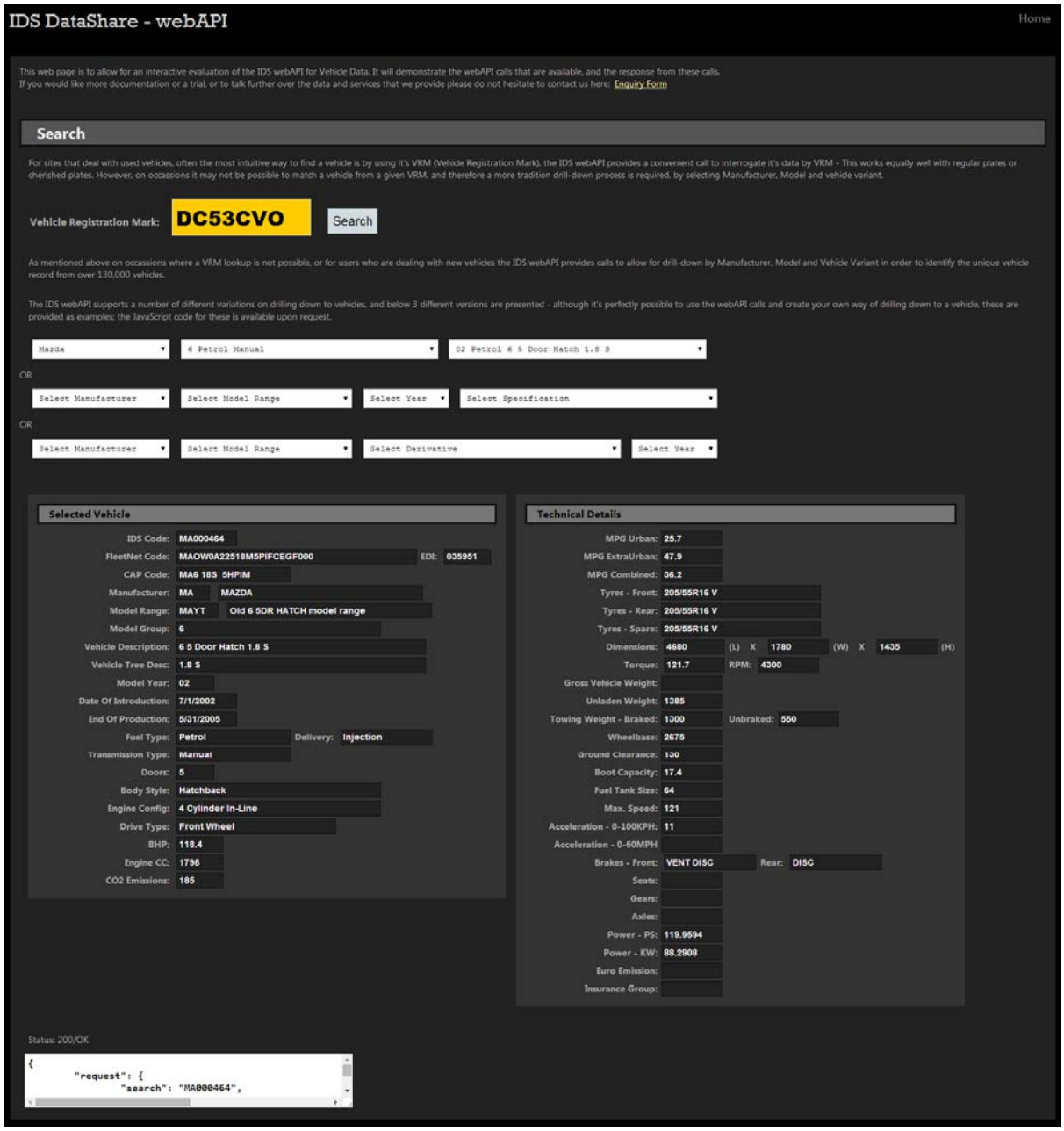
Servers

IDS will maintain two public facing web servers, one is a LIVE server, and will be the server you connect to under normal circumstances – this will always serve the latest vehicle data, and will have the most stable code.

The second server is a TEST server, it's data may not be the very latest (it will not be too far behind, and whilst no active testing is being conducted will also be up to date), the second server will primarily be used to showcase and test new functionality – and where our customers will be invited to test any changed functionality against.

Both servers under normal circumstances will be 100% available for business use during normal business hours; during the early hours of the morning between 3am-6am of each business day service availability may be limited or even none-functioning, as our servers will be having the daily data upgrades.

IDS's vehicle data currently contains over 125,000 vehicles, and we are constantly bringing new vehicles online, updating information on vehicles as soon as manufacturers make us aware of this data; and our web service should also serve up the latest and most current data – we aim to have this performed when most UK businesses are closed, and so when your dealers sign on and start to make use of our web services the following morning, new data is available.



The ‘TEST’ server (at the time of writing this document) is showing off new features, showing 3 different methods of drilling down to a vehicle.

Depending upon your needs, your sites design you may choose to follow any of these drill-down methods, feel free to inspect the code behind the web page; or if your needs are different use the various webAPI calls to construct your own drill-down methods.

API Calls

This list is not exhaustive, and is likely to get added to over time. Some API calls may not be available, due to your subscription model. In the URL's specified below please replace {server} with the server or IP address you are advised to use. Where return data is documented below, it's shown in JSON format – take into account that data can be received in various other formats via content negotiation.

&CustomerId=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

If you are not using jQuery, you may be able to send the Customer Id as part of a customer request header like this:

X-CustomerId : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

This would be preferred however, at the time of writing this document there appears to be an issue when sending the X-CustomerId custom header via jQuery to remote server (seems to work fine on our own server) – hopefully this will be resolved shortly.

We may also adopt IP-Address security, in a way to aid authentication – but again at the time of writing we have not developed this.

All calls must be via **HTTPS** for added security.

{server}/Datashare/api/vehicle?VRM=xxxxxxxx

Search for a vehicle by VRM (where xxxxxxxx is the actual VRM), expected status codes are 200 – vehicle was found, 404 – vehicle was not found.

Vehicle may not be found for a number of reasons, and in these cases you should use dropdowns to prompt the user to try to identify the vehicle by, manufacturer, model and variant.

If the vehicle is found, initial vehicle data is returned to help identify the vehicle matched the format of the return data will be:

```
{
	“request”:	{
			“search”:  //search term used
			“type”:   // type of search performed ‘VRM’
			},
	“data”:		{
			“dateOfVehicleRegistration”:
			“dvla”:		{
						“dateOfManufacture”:
						“dateOfV5C”:
						“vehicleIdentificationNumber”:
						“vehicleIdentificationNumberEnding”:
						“engineNumber”:
						“vehicleColour”:
						“lastColourDate”:
						“previousColours”:
						“priorUse”:
						“manufacturer”:
						“model”:
						“body”:
						“bodyClass”:
						“cherishedTransferMarker”:
						},
			“mvris”:	{
						“manufacturer”:
						“model”:
						“modelVariantName”:
						“engineSize”:
						“cc”:
						“bhp”:
						“doors”:
						“body”:
						“cabType”:
						“gearboxType”:
						“axles”:
						“fuel”:
						},
			“idscode”:	// 8 digit IDS vehicle code
			“fleetnetCode”:
			“fleetnetEdi”:
			“capCode”:
			“manufacturerCode”:
			“manufacturerName”:
			“modelCode”:
			“modelDescription”:
			“modelGroup”:
			“vehicleDescription”:
			“vehicleTreeDescription”:
			“modelYear”:
			“dateOfIntroduction”:
			“endOfProduction”:
			“fuelType”:
			“fuelDelivery”:
			“transmissionType”:
			“doors”:
			“bodyStyle”:
			“engineConfigurationType”:
			“driveTpe”:
```

```
                    “bhp”:
                    “cc”:
                    “co2emissions”:
                },
        “links”: {
                    “technicalUrl”:
                    “serviceUrl”:
                    “imageUrl”:
                },
        “status”: {
                    “code”:
                    “statusText”:
                }
}
```

As you will see from the above response data is essentially made of four parts:

- The ‘request’ object which summarises what was just requested and how that search was satisfied.
- The ‘data’ object which will contain all of the relevant data for the search. In the case of a VRM search it contains the found vehicle, and sufficient fields to identify the vehicle – perhaps display on a web page for confirmation by the user.
- The ‘links’ object which will contain any relevant URL’s for further information on this vehicle.
- The ‘status’ object which contains HTTP ‘code’ and ‘statusText’ for the success or failure of the call – this object should ALWAYS be inspected, status.code = 200 = success.

All web API calls will return response data with these four named parts – ‘request’, ‘data’ ‘links’ and ‘status’ – though the content of each part will be relevant to the type of call being made.

NOTE: FleetNet & CAP field in the above returned data will only be populated if you have the appropriate FleetNet and CAP licenses.

NOTE: DVLA and MVRIS sections will only return data in individual fields if enabled in your subscription.

“idscode” field is the unique identity of the vehicle within IDS data, and is therefore the key code that will need to be used for other calls to obtain more information, such as service information or technical information about this individual vehicle.

VRM searches may fail for a variety of reasons, here are a few to consider:

- Vehicle is too old – IDS maintain vehicle data upto 10yrs old, if the vehicle is older than this, then we are unable to match the VRM to an IDS vehicle.
- Vehicle is too new – With extremely newly registered vehicles, it may well be that the DVLA have not updated their records, or our VRM lookup supplier hasn’t managed to update their records, and therefore we have no records to match off against an IDS vehicle. You will find however, that if you drilldown through the IDS vehicles, we will be maintaining this vehicle, our vehicle data is up to date, as we are advised many months before a vehicle goes onto release.
- Bad records – there are a few instances where bad record keeping by the DVLA has resulted in failed searches – each VRM search iterates through a number of different linked databases held by the DVLA, MVRIS and CDL – if a link is broken in any one of these systems, then the vehicle cannot be matched.

All said though, the VRM matching process is extremely quick and accurate and aims to match a high proportion of vehicles by their VRM.

In instances where a VRM search does not find a vehicle, the user will need to be prompted via a series of drop downs, selecting manufacturer, model and vehicle variant, the following 3 web API calls will allow you to achieve this.

{server}/Datashare/api/manufacturers

This will return a list of manufacturers, and their associated manufacturer 2-digit codes.

Optional parameters to this call are:

- **pageSize=nn** - This allows you to limit the maximum number of results returned
- **page=nn** - This in combination with pageSize allows you to select a specific page of results – these two parameters are common on API calls where multiple results can be received.

Data will be returned in the following format:

```
{
	“request”: {
				“search”:  //search term used
				“type”:   // type of search performed ‘Manufacturers’
		},
	“data”: {
			“manufacturers”: // array of manufacturers
			{
				“manufacturerCode”:
				“manufacturerName”:
				“modelsUrl”:
			}
		},
	“links”: {
			“nextPage”:
			“prevPage”:
		},
	“status”: {
			“code”:
			“statusText”:
		}
}
```

The nextPage & prevPage links provide convenient URL’s to navigate to the next/previous page of manufacturers. Though for the quantity of manufacturers it would be likely all manufacturers would be requested in one hit. If there isn’t a previous page (because either all data was requested, or this is the first page) then the prevPage url will be empty/null. The same is true for nextPage.

Within ‘data’ object this will contain a list of ‘manufacturers’, an array of manufacturer object – which include the manufacturers 2-digit code, it’s name that will be displayed to the user, and a URL that can be used to list the models of that manufacturer. Note: The modelsURL will not include any paging information or customerID’s; so these will need to be appended in the normal way.

{server}/Datashare/api/models?manufacturer=xx

This will return a list of Models for a given manufacturer – where xx is the 2-digit manufacturer code.

Optional parameters to this call are:

- **pageSize=nn** - This allows you to limit the maximum number of results returned
- **page=nn** - This in combination with pageSize allows you to select a specific page of results.

Data will be returned in the following format:

```
{
	“request”: {
				“search”:  //search term used
				“type”:   // type of search performed ‘Models’
		},
	“data”: {
			“models”: // array of models
			{
				“modelCode”:
				“modelDescription”:
				“vehiclesUrl”:
				“model”:
				“fuelType”:
				“transmissionType”:
			}
		},
	“links”: {
			“nextPage”:
			“prevPage”:
		},
	“status”: {
			“code”:
			“statusText”:
		}
}
```

Within ‘data’ object this will contain a list of ‘models’, an array of model object – which include the model 2-digit code, it’s name that will be displayed to the user, and a URL that can be used to list the vehicles within that model.

Model descriptions that are returned here are essentially a ‘composite’ of Model + fuelType + transmissionType.

{server}/Datashare/api/vehicles?model=xxxx

This will return a list of vehicles for the given model code. Depending upon the model this could be a list of several hundred vehicles; so it is probably ideal to page this data.

Optional parameters to this call are:

- **pageSize=nn** - This allows you to limit the maximum number of results returned
- **page=nn** - This in combination with pageSize allows you to select a specific page of results.

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘Vehicles’
			},
	“data”:		{
				“vehicles”:  // array of vehicles
				{
					“idscode”:   // 8 digit IDS vehicle code
					“fleetnetCode”:
					“fleetnetEdi”:
					“capCode”:
					“manufacturerCode”:
					“manufacturerName”:
					“modelCode”:
					“modelDescription”:
					“modelGroup”:
					“vehicleDescription”:
					“modelYear”:
					“dateOfIntroduction”:
					“endOfProduction”:
					“fuelType”:
					“fuelDelivery”:
					“transmissionType”:
					“doors”:
					“bodyStyle”:
					“engineConfigurationType”:
					“driveTpe”:
					“bhp”:
					“cc”:
					“co2emissions”:
					“links”: {
								“technicalUrl”:
								“serviceUrl”:
								“imageUrl”:
							}

				}
			},
	“links”: {
				“nextPage”:
				“prevPage”:
			},
	“status”: {
				“code”:
				“statusText”:
			}
}
```

Within the “data” object there is an array of “vehicles”, each vehicle object contains relevant vehicle codes and data to help identify the vehicle. Each individual vehicle returned will also contain various links for further information on the vehicle.

{server}/Datashare/api/services?idscode=xxxxxxxx

Retrieve Servicing information for the specified vehicle. This service data is RAW service data

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘Services’
			},
	“data”:		{
				“idscode”:	// 8 digit IDS vehicle code
				“firstServiceMileage”:
				“firstServiceMonths”:
				“initialServiceMileage”:
				“initialServiceMonths”:
				“subsequentServiceMileage”:
				“subsequentServiceMonths”:
				“firstServiceMileage2”:
				“firstServiceMonths2”:
				“initialServiceMileage2”:
				“initialServiceMonths2”:
				“subsequentServiceMileage2”:
				“subsequentServiceMonths2”:
				“milesToRegime2”:
				“oilCapacity”:
				“labourRate”:
				“serviceIndicator1”:
				“serviceIndicator2”:
				“serviceRegimes”: // 1 or 2 service regimes
				{
					“regime”:	// 1 or 2
					“services”:  // array of services
					{
						“mileage”:
						“partsCost”:
						“labourTime”:
					}
					“serviceItems”:  // array of service items
					{
						“mileage”:
						“months”:
						“serviceItemCode”:
						“serviceItemDescription”:
						“serviceItemIncludedInCost”:
						“labourTime”:
						“itemCost”:
						“everyTime”:
						“type”:
					}

				}


		},
	“links”: {
				“vehicleUrl”:
				“technicalUrl”:
				“imageUrl”:
		},
	“status”: {
				“code”:
				“statusText”:
			}
}
```

{server}/Datashare/api/vehicle?idscode=xxxxxxxx

Search for a vehicle by IDS code (where xxxxxxxx is the actual IDS code), expected status codes are 200 – vehicle was found, 404 – vehicle was not found.

If the vehicle is found, initial vehicle data is returned to help identify the vehicle matched, the format of the return data will be:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘Vehicle’
			},
	“data”:		{
			“idscode”:   // 8 digit IDS vehicle code
			“fleetnetCode”:
			“fleetnetEdi”:
			“capCode”:
			“manufacturerCode”:
			“manufacturerName”:
			“modelCode”:
			“modelDescription”:
			“modelGroup”:
			“vehicleDescription”:
			“vehicleTreeDescription”:
			“modelYear”:
			“dateOfIntroduction”:
			“endOfProduction”:
			“fuelType”:
			“fuelDelivery”:
			“transmissionType”:
			“doors”:
			“bodyStyle”:
			“engineConfigurationType”:
			“driveTpe”:
			“bhp”:
			“cc”:
			“co2emissions”:
		},
	“links”: {
			“technicalUrl”:
			“serviceUrl”:
			“imageUrl”:
		},
	“status”: {
			“code”:
			“statusText”:
		}
}
```

{server}/Datashare/api/technical?idscode=xxxxxxxx

Retrieve all technical vehicle data for a given vehicle (using it's IDS code).

Data will be returned in the following format:

```
{
	"request":	{
				"search":  //search term used
				"type":   // type of search performed 'Technical'
			},
	"data":		{
			"idscode":   // 8 digit IDS vehicle code
			"mpgUrban":
			"mpgExtraUrban":
			"mpgCombined":
			"torque":
			"torqueRpm":
			"length":
			"width":
			"height":
			"grossVehicleWeight":
			"unladenWeight":
			"towingWeightBraked":
			"towingWeightUnbraked":
			"wheelbase":
			"groundClearance":
			"bootCapacity":
			"tankSize":
			"maximumSpeed":
			"acceleration0to100kph":
			"accelleration0to60mph":
			"frontBrakes":
			"rearBrakes":
			"frontTyres":
			"rearTyres":
			"spareTyre":
			"seats":
			"gears":
			"axles":
			"powerPS":
			"powerKW":
			"euroEmission":
			"insuranceGroup":
		},
	"links": {
			"vehicleUrl":
			"serviceUrl":
			"imageUrl":
		},
	"status": {
			"code":
			"statusText":
		}
}
```

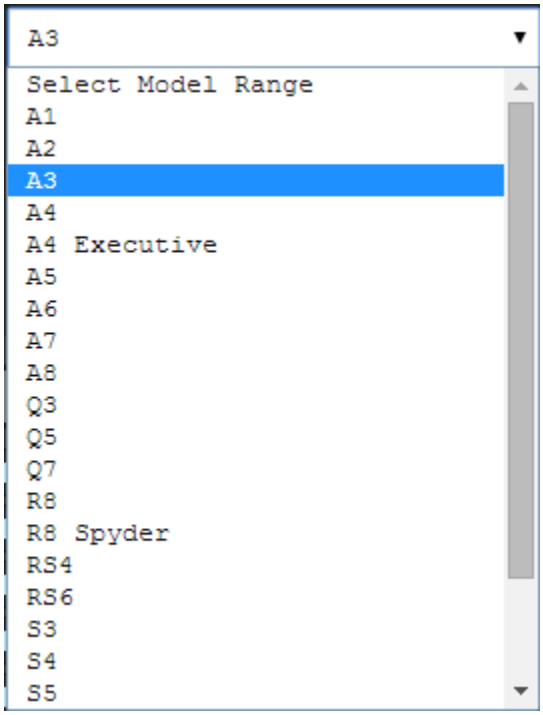
{server}/Datashare/api/modelranges?manufacturer=XX

Retrieve a list of Model Ranges for the specified Manufacturer (by model ranges, this is ‘A3’, ‘A4’, ‘A5’, ‘A6’, etc for Audi as an example)

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘ModelRanges’
			},
	“data”:		{
			modelRanges:{ //Array of ModelRanges
				“modelRangeCode”:
				“modelRangeDescription”:
				}
		},
	“links”: {
			“nextPage”:
			“prevPage”:
		},
	“status”: {
			“code”:
			“statusText”:
		  }
}
```

This call can be seen on the sample web page, within the 2nd drill-down method:



{server}/Datashare/api/modelyears?modelRangeCode=nn

Retrieve a list of Years for a selected Model Range.

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘ModelYears’
			},
	“data”:		{
			modelYears:{ //Array of ModelYears
				“yearCode”:
				“year”:
				}
		},
	“links”: {
			“nextPage”:
			“prevPage”:
		},
	“status”: {
			“code”:
			“statusText”:
		}
}
```

This call can be seen on the sample web page, within the 2nd drill-down method:

{server}/Datashare/api/vehicles?modelYearCode=xxxx-xxxx

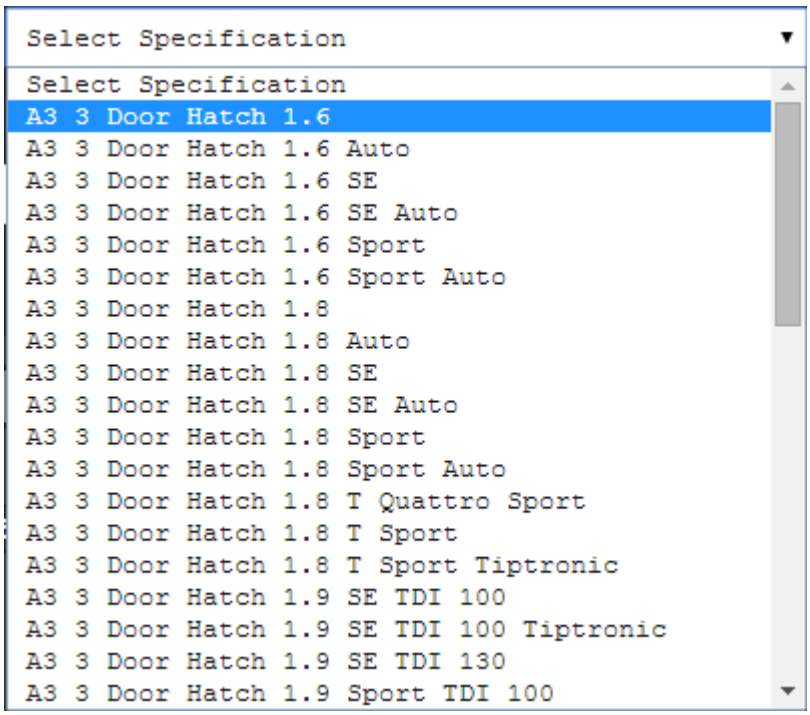
Retrieve a list of vehicles for a given modelYearCode – this will return an array of vehicles, with sufficient information to identify the required vehicle from.

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘Vehicles’
			},
	“data”:		{
				“vehicles”:  // array of vehicles
				{
					“idscode”:   // 8 digit IDS vehicle code
					“fleetnetCode”:
					“fleetnetEdi”:
					“capCode”:
					“manufacturerCode”:
					“manufacturerName”:
					“modelCode”:
					“modelDescription”:
					“modelGroup”:
					“vehicleDescription”:
					“modelYear”:
					“dateOfIntroduction”:
					“endOfProduction”:
					“fuelType”:
					“fuelDelivery”:
					“transmissionType”:
					“doors”:
					“bodyStyle”:
					“engineConfigurationType”:
					“driveTpe”:
					“bhp”:
					“cc”:
					“co2emissions”:
					“links”: {
								“technicalUrl”:
								“serviceUrl”:
								“imageUrl”:
							}

				}
			},
	“links”: {
				“nextPage”:
				“prevPage”:
			},
	“status”: {
				“code”:
				“statusText”:
			}
}
```

This call can be seen on the sample web page, within the 2nd drill-down method:



For the purpose of this dropdown, I'm using the vehicleDescription property.

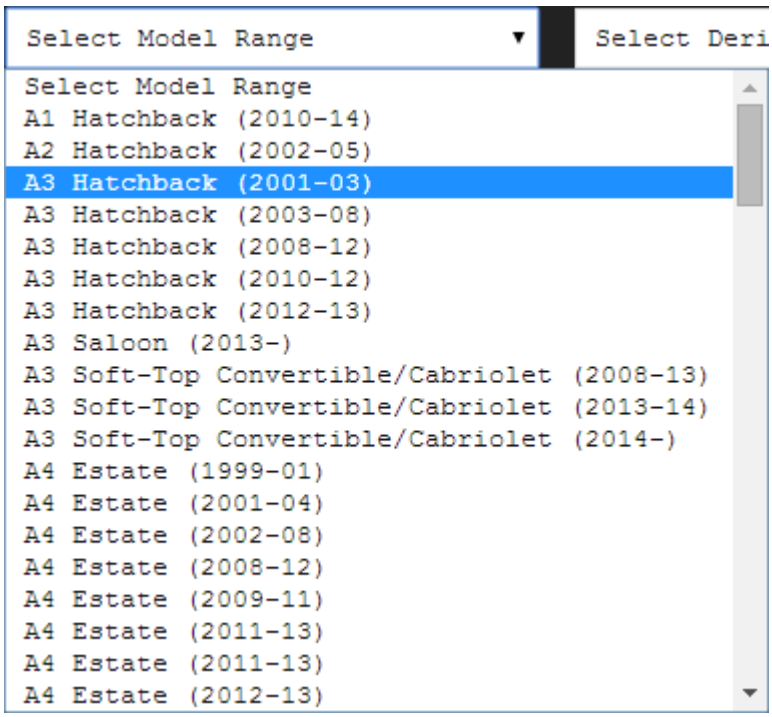
{server}/Datashare/api/modelyearranges?manufacturer=XX

Retrieve a list of Model Year Ranges for the specified manufacturer.

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘ModelYearRanges’
			},
	“data”:		{
			modelYearRanges:{   // Array of ModelYearRanges
				“modelYearRangeCode”:
				“modelYearRange”:
				}
		},
	“links”: {
			“nextPage”:
			“prevPage”:
		},
	“status”: {
			“code”:
			“statusText”:
		}
}
```

This call can be seen on the sample web page, within the 3rd drill-down method:



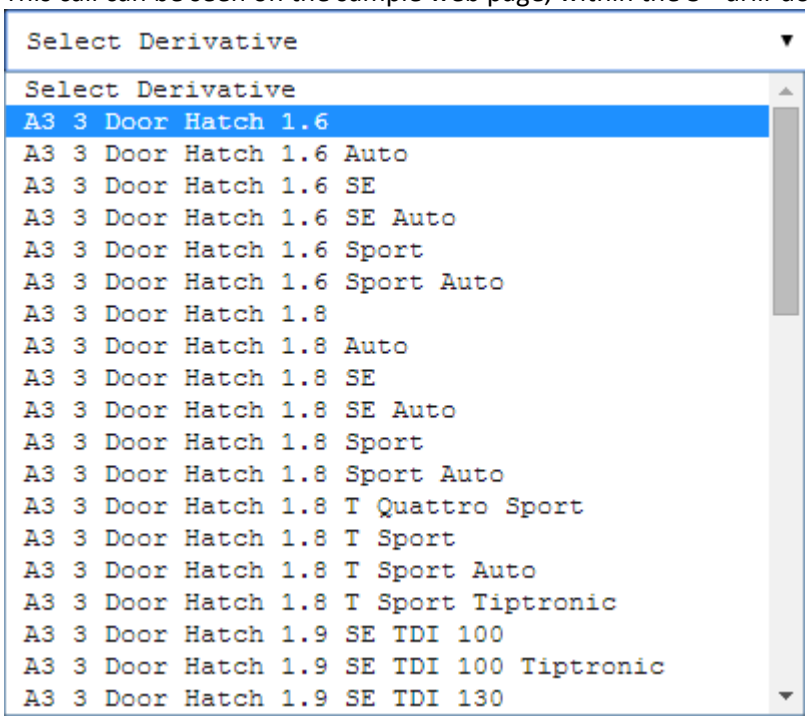
{server}/Datashare/api/modelderivatives?modelYearRangeCode=xxxxx

Retrieve a list of Model Year Derivatives for the specified modelYearRangeCode.

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘ModelDerivatives’
			},
	“data”:		{
			modelDerivatives:{  // Array of ModelDerivatives
				“derivativeCode”:
				“derivativeDescription”:
				}
		},
	“links”: {
			“nextPage”:
			“prevPage”:
		},
	“status”: {
			“code”:
			“statusText”:
		}
}
```

This call can be seen on the sample web page, within the 3rd drill-down method:



{server}/Datashare/api/vehicles?derivativeCode=xxxxxx

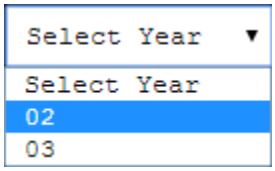
Retrieve a list of vehicles for the specified DerivativeCode.

Data will be returned in the following format:

```
{
	“request”:	{
				“search”:  //search term used
				“type”:   // type of search performed ‘Vehicles’
			},
	“data”:		{
				“vehicles”:  // array of vehicles
				{
					“idscode”:    // 8 digit IDS vehicle code
					“fleetnetCode”:
					“fleetnetEdi”:
					“capCode”:
					“manufacturerCode”:
					“manufacturerName”:
					“modelCode”:
					“modelDescription”:
					“modelGroup”:
					“vehicleDescription”:
					“modelYear”:
					“dateOfIntroduction”:
					“endOfProduction”:
					“fuelType”:
					“fuelDelivery”:
					“transmissionType”:
					“doors”:
					“bodyStyle”:
					“engineConfigurationType”:
					“driveTpe”:
					“bhp”:
					“cc”:
					“co2emissions”:
					“links”: {
								“technicalUrl”:
								“serviceUrl”:
								“imageUrl”:
							}

				}
			},
	“links”: {
				“nextPage”:
				“prevPage”:
			},
	“status”: {
				“code”:
				“statusText”:
			}
}
```

This call can be seen on the sample web page, within the 3rd drill-down method:



For the purpose of this dropdown, I’m using the modelYear property.