



DataOne Software

Providing Powerful Data Solutions to the Automotive Marketplace

XML VIN Decoder, v6.0

Product Usage Guide

April, 2009

1.0 XML VIN Decoder v6 Overview

The DataOne Software XML VIN Decoder is a simple REST-based web service that can be used to obtain detailed and precise vehicle information when given a 17-digit Vehicle Identification Number (VIN). Parameters are supplied by the client via an HTTP GET string, and results are returned in a simple, easy to parse XML format. This document will explain in detail all aspects of the web service, from input parameters to available data packs to a detailed listing of the XML syntax.

The XML VIN Decoder is powered by DataOne's comprehensive Light Duty Vehicle US-Market database, which covers all vehicles sold in the US consumer market from 1981 to present. Additional datapacks are available for Motorcycles, Powersports, and limited medium-duty and heavy-duty vehicles. Please contact your sales representative if you are interested in learning more about this data.

2.0 Available Data

The XML VIN Decoder goes far beyond the standard data that is encoded directly into a VIN Number by the OEM, which generally includes Year, Make, Model, Engine, Transmission, Gross Vehicle Weight Range (GVWR), and Restraint Type. DataOne offers the following VIN-based data via the XML VIN Decoder:

- ❖ Basic vehicle descriptors
- ❖ Interior and Exterior Vehicle dimensions
- ❖ Technical and Performance Specifications
- ❖ Engines
- ❖ Transmissions
- ❖ Fuel Economy
- ❖ Features
- ❖ Options
- ❖ Colors
- ❖ Warranties
- ❖ Pricing
- ❖ Stock Photos

Additional data types are available for certain markets, including awards & accolades data, and premium imagery and video media. Please contact your sales representative for more details, or to have additional data enabled for your account.

3.0 Interacting with the Decoder

Your application can access data from the XML VIN Decoder by making a simple HTTP GET request, using the URL and credentials supplied to you in your free trial welcome letter. Alternatively, you can access the service directly through a web browser to view the raw XML output. This can be especially useful when debugging integration problems. Most browsers, including Mozilla Firefox and Internet Explorer, will format the XML structure in an easily viewable format. Several basic GET query string parameters are necessary to perform a decode, and additional optional parameters can be used to narrow or expand the data return.

Parameter	Explanation	Usage Example
client_id	Your client ID, as specified in your free trial welcome letter.	&client_id=99999999
auth_code	Your authorization code, as specified in your free trial welcome letter	&auth_code=gh83243NsadfF3234kdsaf
vins	Either a single 17-digit VIN Number, or a comma-separated list of as many as 50 VINs. Spaces should not be added between VINs.	&vins=2T3ZF33V09W003746 -or- &vins=2T3ZF33V09W003746,1FTDX0864VKA85619
Example URL: http://www.xmlvindecoder.com/rest/decoder.php?client_id=99999999&auth_code=gh83243NsadfF3234kdsaf&vins=2T3ZF33V09W003746,1FTDX0864VKA85619		

Table 3.1 – Required XML VIN Decoder v6 Parameters

The above table shows the three parameters that you must pass in your GET query string for each decode. Note that the sample URL has each parameter and value pair separated by an ampersand (&), with the first parameter preceded by a question mark (?). Those who are familiar with GET query strings should recognize this format. An equal sign separates parameter names from values. The order of these parameters in the query string does not matter, however it is important that the first and only the first name-value pair is preceded by a question mark.

TIP: You'll need to replace the dummy values with your own for client_id and auth_code

TIP: For more information on GET strings, check out: http://en.wikipedia.org/wiki/Query_string

3.1 Multiple Styles

The optional parameters are designed to help with cases where a single VIN could be referencing any one of several *vehicle styles*. A vehicle style is a unique representation of a single year, make, model, trim, body type, and occasionally drive type. Many manufacturers encode the exact vehicle style into the VIN number, while others encode only a partial description. In these cases, you would generally need to select one of several styles to obtain complete vehicle data.

Fortunately, DataOne’s XML VIN Decoder provides a number of features that help both with style selection, and with obtaining usable data even when a specific style cannot be determined. A full listing of data that is common or shared across multiple styles is returned with each VIN, and a number of parameters are available to use for style selection.

Parameter	Explanation	Usage Example
style_ids	Either a single available vehicle style ID, or multiple comma separated IDs. If this parameter is used, the number of vehicle style IDs supplied must match the number of VINs.	&style_ids=400869481 -or- &style_ids=400869481,9474
model_numbers	Either a single manufacturer model number, or multiple comma separated model numbers. If this parameter is used, the number of model numbers must match the number of VINs.	&model_numbers=4430 -or- &model_numbers=4430,502A
show_style_loop	A flag indicating that VINs that map to multiple vehicle styles will or will not have the data for each style exploded.	&show_style_loop=0 (default) -or- &show_style_loop=1
<p>Example URLs:</p> <p>http://www.xmlvindecoder.com/rest/decoder.php?client_id=999999999&auth_code=gh83243NsadfF3234kdsaf&vins=1FTDX0864VKA85619&style_ids=9474</p> <p>http://www.xmlvindecoder.com/rest/decoder.php?client_id=999999999&auth_code=gh83243NsadfF3234kdsaf&vins=2T3ZF33V09W003746,1FTDX0864VKA85619&model_numbers=4430,502A</p> <p>http://www.xmlvindecoder.com/rest/decoder.php?client_id=999999999&auth_code=gh83243NsadfF3234kdsaf&vins=2T3ZF33V09W003746,1FTDX0864VKA85619&show_style_loop=1</p>		

Table 3.2 – Optional XML VIN Decoder v6 Parameters

style_ids

The XML return for each VIN decode contains both a <common_data> section, containing data that is common across all available styles, and an <available_vehicle_styles> section. In cases where a VIN is mapped to only a single vehicle style, the <available_vehicle_styles> section will contain only one vehicle style. A full vehicle data listing will appear under the <vehicle_style> tag for this vehicle style, and there will be no need to resubmit your request using the style_ids parameter.

In cases where a VIN is mapped to multiple vehicle styles, a list of empty <vehicle_style> tags are provided. Each <vehicle_style> tag contains both a name and style_id attribute. The name attribute contains the name of the style which, when combined with year, make, model, and trim from the common data section, will uniquely identify the vehicle style. The style_id attribute contains an ID number that can be passed back to the XML VIN Decoder service as the value of the style_ids parameter in a second call to retrieve all of the information unique to this specific vehicle style.

By default, the process of decoding a VIN that could represent several vehicle styles is a two-step process. However, the second step can be avoided with one of the next two optional parameters.

model_numbers

In many industries, it is common to have access to both a VIN and a manufacturer model number. Manufacturer model numbers are often used by OEMs to uniquely identify a specific vehicle style within a given year and model. The combination of a VIN and a model number are almost always sufficient to fully decode a vehicle. You can pass manufacturer model numbers in your GET query using the model_numbers parameter. Comma separated values in the vins and model_numbers attributes are mapped in a one-to-one fashion.

If you include two VINs and two model numbers, the first model number is used with the first VIN and the second with the second. If you have several VINs and wish not to include a model number for one of them, simply leave the portion of the comma separated model number string that would contain that VIN's model number blank. For example:

```
&vins=2T3ZF33V09W003746,1GYEE637760133330,1FTDX0864VKA85619  
&model_numbers=4430,,502A
```

show_style_loop

Setting show_style_loop=1 will cause the empty <vehicle_style> tags in sets of multiple vehicle styles to populate with full data for each style. This is set to 0 by default to minimize the size of the XML return for VINs with multiple styles, but you may find it appropriate to run some or all of your requests with this parameter set to 1.

4.0 The XML Syntax

This section details the syntax of the XML VIN Decoder's XML return. A separate Data Dictionary is available on the client section of our website that defines each individual element and attribute.

```
<decoded_vin_data>
  <decoded_version>6.0.x</decoded_version>
  <decoder_error>
    [Decoder-level Error Section]
  </decoder_error>
  <vin_number value="">
    <vin_error>
      [VIN-level Error Section]
    </vin_error>
    <common_data>
      [Common Data Section]
    </common_data>
    <available_vehicle_styles>
      <vehicle_style name="" style_id="" complete="">
        [Vehicle Style Section]
      </vehicle_style>
      ...
    </available_vehicle_styles>
  </vin_number>
  ...
</decoded_vin_data>
```

Figure 4.1 – A high-level view of the XML VIN Decoder Syntax

Here you can see the root XML element is *decoded_vin_data*, which contains all of the XML in the return. Within this element there are elements for the decoder version, any decoder-level errors that have occurred, and one or more *vin_number* elements, for each VIN supplied in the *vins* parameter.

Each *vin_number* element contains a *vin_error* element, which lists any VIN-level errors that have occurred, as well as a *common_data* section showing all of the data that is shared across multiple vehicle styles, and an *available_vehicle_styles* section. If only one *vehicle_style* is associated with the VIN in question, or if a *style_id* or valid *model_number* have been passed along with the VIN, this section will contain only one *vehicle_style* element. If the VIN is associated with multiple vehicle styles and a single vehicle style could not be chosen based on the *model_number* and *style_id* parameters, it will contain multiple *vehicle_style* elements. Whether or not multiple *vehicle_style* elements contain data will depend on the *show_style_loop* parameter, which defaults to 0 (data is not shown).

4.1 Decoder-level Errors

The *decoder_error* element will contain two sub-elements if a Decoder-level error occurs: *error_code* and *error_message*. These will be populated with the following codes and messages in the event that your account is currently unable to decode VINs. You should contact our sales or support department if you receive one of these messages.

Sales & Support Contact Information:

Sales E-mail: sales@dataonesoftware.com

Support E-mail: datasupport@dataonesoftware.com

Phone: 1-877-GET-VINS

Code	Message
SP	Access denied. Service is suspended.
ET	Access denied. Trial ended on [DATE].
ED	Access denied. Maximum decodes has been reached for your trial.

Table 4.1 – Decoder-level Errors and Codes

4.2 VIN-level Errors

The *vin_error* element will contain two sub-elements if a VIN-level error occurs: *error_code* and *error_message*. These will be populated with the following codes and messages in the event that the supplied VIN could not be decoded.

Code	Message	Explanation
IV	Invalid VIN: Not 17 characters Invalid VIN: Failed Checksum Invalid VIN: Invalid Engine Code Invalid VIN: Invalid Plant Digit Invalid VIN: Invalid Restraint Digit Invalid VIN: Invalid VDS Invalid VIN: Invalid WMI	The XML VIN Decoder performs a number of checks on each VIN submitted to ensure it is indeed a valid VIN. If the decoder determines the VIN is invalid, an error message IV is thrown along with a description of the specific reason why the VIN is invalid.
MV	You cannot decode more than 50 VINs at a time	The XML VIN Decoder has a 50-VIN per request limit.
CS	The VIN and Style ID combination you provided seems to be invalid	The <i>style_id</i> provided for this VIN does not exist.

AT	This VIN could not be decoded. It appears to be an ATV.
PR	This VIN could not be decoded. It appears to be a pre-1981 vehicle.
CA	This VIN could not be decoded. It appears to be a Canadian vehicle.
CH	This VIN could not be decoded. It appears to be a Chassis vehicle.
FL	This VIN could not be decoded. It appears to be a Fleet vehicle.
GM	This VIN could not be decoded. It appears to be a Grey Market vehicle.
HV	This VIN could not be decoded. It appears to be a Heavy-duty vehicle.
MD	This VIN could not be decoded. It appears to be a Mid-duty vehicle.
TR	This VIN could not be decoded. It appears to be a Trailer.
OM	This VIN could not be decoded. It appears to be an Out Of Market vehicle.
UM	This VIN could not be decoded. It appears to be a Motorcycle.
RV	This VIN could not be decoded. It appears to be a Recreational Vehicle.
UK	This VIN could not be decoded. It has been added to our research queue.

Table 4.2 – VIN-level Errors and Codes

4.3 Common Data Section

The common data element contains a number of sub-elements, populated with data that is shared across all of the vehicle styles associated with the VIN listed in its parent element. The syntax is an exact copy of the Vehicle Style section, but with the media section removed.

4.4 Vehicle Style Section

The vehicle style section, contained within a *vehicle_style* element, is a listing of all of the data associated with the named vehicle style. It can include all of the data types listed on page one of this document, although certain elements may not be populated depending on your specific account settings. You should examine the specific XML return made available to your account, as tags that have been deactivated for your account may appear as empty elements, or simply may not be included at all.

The following page shows a complete listing of the XML syntax for the *vehicle_style* element, followed by a detailed view of each major sub-element.

```

<vehicle_style name="" style_id="" complete="">
  <basic_data>
    [Basic Data Section]
  </basic_data>
  <specifications>
    [Specifications Section]
  </specifications>
  <engines>
    [Available Engines section]
  </engines>
  <transmissions>
    [Available Transmissions section]
  </transmissions>
  <fuel_efficiency_ratings>
    [Fuel efficiency section]
  </fuel_efficiency_ratings>
  <eng_trans_mpg_associations>
    [Engine, Transmission, MPG Association section]
  </eng_trans_mpg_associations>
  <features>
    [Features section]
  </features>
  <options>
    [Options section]
  </options>
  <colors>
    [Colors section]
  </colors>
  <warranties>
    [Warranties section]
  </warranties>
  <pricing>
    [Pricing section]
  </pricing>
  <media>
    [Media section]
  </media>
</vehicle_style>

```

Figure 4.2 – XML Syntax for the *vehicle_style* element group

TIP: A data dictionary is available on the Client Section of our website. It contains detailed definitions for each datapoint, and in many cases includes sample data.

4.4.1 Basic Data Section

```
<basic_data>
  <year></year>
  <make></make>
  <model></model>
  <trim></trim>
  <style></style>
  <vehicle_type></vehicle_type>
  <body_type> </body_type>
  <body_subtype></body_subtype>
  <doors></doors>
  <model_number></model_number>
  <package_code></package_code>
  <country_of_manufacture></country_of_manufacture>
  <plant></plant>
</basic_data>
```

Figure 4.3 – XML Syntax for the *basic_data* element group

4.4.2 Specifications Section

```
<specifications>
  <category name="Driven Wheels">
    <drive_type>4WD</drive_type>
    <hub_type_4wd/>
  </category>
  <category name="Performance">
    <acceleration_to_100></acceleration_to_100>
    <acceleration_to_60></acceleration_to_60>
    <aerodynamic_drag></aerodynamic_drag>
    <braking_distance></braking_distance>
    <turning_circle></turning_circle>
  </category>
  <category name="Size and Shape Measurements">
    <angle_of_approach></angle_of_approach>
    <angle_of_departure></angle_of_departure>
    <breakover_angle></breakover_angle>
    <front_track></front_track>
    <ground_clearance></ground_clearance>
    <height></height>
    <length></length>
    <length_no_bumpers></length_no_bumpers>
    <rear_track></rear_track>
    <wheelbase></wheelbase>
    <width></width>
    <width_no_mirrors></width_no_mirrors>
  </category>
```

(Continued on following page)

```

<category name="Weight Measurements">
  <curb_weight></curb_weight>
  <gross_combined_weight_rating></gross_combined_weight_rating>
  <gross_vehicle_weight_rating></gross_vehicle_weight_rating>
  <gross_vehicle_weight_range></gross_vehicle_weight_range>
  <max_payload></max_payload>
  <max_towing_capacity></max_towing_capacity>
  <base_towing_capacity></base_towing_capacity>
</category>
<category name="Interior Dimensions">
  <cargo_volume></cargo_volume>
  <cargo_volume_rear_seats_down></cargo_volume_rear_seats_down>
  <cargo_volume_row3_down></cargo_volume_row3_down>
  <interior_volume></interior_volume>
  <passenger_volume></passenger_volume>
  <passenger_volume_third_row></passenger_volume_third_row>
</category>
<category name="Seating">
  <head_room_front></head_room_front>
  <head_room_rear></head_room_rear>
  <head_room_third_row></head_room_third_row>
  <hip_room_front></hip_room_front/>
  <hip_room_rear></hip_room_rear>
  <hip_room_thrid_row></hip_room_third_row>
  <leg_room_front></leg_room_front>
  <leg_room_rear></leg_room_rear>
  <leg_room_third_row></leg_room_third_row>
  <max_seating></max_seating>
  <seating_rows></seating_rows>
  <shoulder_room_front></shoulder_room_front>
  <shoulder_room_rear></shoulder_room_rear>
  <shoulder_room_third_row></shoulder_room_third_row>
  <std_seating></std_seating>
</category>
<category name="Truck Bed">
  <bed_code></bed_code>
  <bed_length></bed_length>
</category>
<category name="Wheels and Tires">
  <rear_tire_type></rear_tire_type>
  <rear_wheel_dia></rear_wheel_dia>
  <tire_type></tire_type>
  <wheel_dia></wheel_dia>
</category>
<category name="Fuel Storage">
  <tank_1_capacity></tank_1_capacity>
  <tank_2_capacity></tank_2_capacity>
</category>
</specifications>

```

Figure 4.4 – XML Syntax for *specifications* element group

4.4.3 Available Engines Section

The engines element group contains detailed information for all engines available for the VIN provided. Vehicles are commonly sold with any one of several available engine options installed. Usually the VIN number uniquely identifies which of the available engines is actually installed on the vehicle, and in these cases the *engines* element contains only one *engine* element. In these cases, the *standard* attribute will be set to “NA” for not applicable, and the *installed* attribute will be set to “Y.”

In some cases, the VIN does not uniquely identify the installed engine, and in this case several engine options will be listed. The “standard” or default engine option will have a “Y” in its *standard* attribute, while the other options will have a “N.” Since the specific engine installed on the vehicle is not known, the *installed* attribute will be set to “UK,” for unknown, for all of the options.

```
<engines>
  <engine brand="" name="" id="" standard="" installed="">
    <aspiration></aspiration>
    <block_type></block_type>
    <bore></bore>
    <cam_type></cam_type>
    <compression></compression>
    <cylinders></cylinders>
    <displacement></displacement>
    <fuel_induction></fuel_induction>
    <fuel_quality></fuel_quality>
    <fuel_type></fuel_type>
    <invoice_price></invoice_price>
    <marketing_name></marketing_name>
    <max_hp></max_hp>
    <max_hp_at></max_hp_at>
    <max_payload></max_payload/>
    <max_torque></max_torque>
    <max_torque_at></max_torque_at>
    <msrp_price></msrp_price>
    <oil_capacity></oil_capacity>
    <order_code></order_code>
    <redline></redline>
    <stroke></stroke>
    <valve_timing></valve_timing>
    <valves></valves>
  </engine>
</engines>
```

Figure 4.5 – XML Syntax for *engines* element group

4.4.4 Available Transmissions Section

Just as vehicles can be sold with any one of several available engines, they are often sold with any one of several available transmissions. Often times the VIN does not uniquely identify which transmission is installed on the vehicle, and so the *transmissions* section of the XML return will often contain multiple available transmission options. As was the case with engines, the default transmission option will be given a value of “Y” for the *standard* attribute in a listing of multiple transmissions, and installed will be set to “UK” for unknown. If a single transmission can be determined from the VIN, it will be marked with a “Y” for the *installed* attribute.

```
<transmissions>
  <transmission brand="" name="" id="" standard="" installed="">
    <type></type>
    <detail_type></detail_type>
    <gears></gears>
  </transmission>
</transmissions>
```

Figure 4.6 – XML Syntax for *transmissions* element group

4.4.5 Fuel Efficiency Section

Fuel efficiency ratings depend on a number of factors. Different trim and package configurations can change the mass of the vehicle, lowering or improving MPG. A more powerful engine will generally have a lower MPG rating, and manual transmissions tend to outperform automatic transmissions. The presence of a locking torque converter vs. a non-locking torque converter can also add an extra MPG or two. A lot needs to be known about the vehicle to get an exact MPG number.

As described in the previous two sections, there are many cases where both the engine and transmission are known for the given VIN number. In these cases, we can usually provide a single city MPG number and a single highway MPG number. In cases where this is not possible, a range is provided for both city and highway. If the VIN submitted could be equipped with either a manual or automatic transmission, separate ranges are supplied for each transmission type. The *trans_type* attribute will contain AT, MT, or NA for automatic, manual, or not applicable (in cases where only a single transmission type is present).

```
<fuel_efficiency_ratings>
  <mpg_city low="" high="" trans_type="">
  <mpg_hwy low="" high="" trans_type="">
</fuel_efficiency_ratings>
```

Figure 4.7 – XML Syntax for *fuel_efficiency_ratings* element group

4.4.6 Engine, Transmission, MPG Association Section

The Engine, Transmission, MPG Association section describes which transmissions are available for which engines, and what the MPG ratings are for the combination of the two. There are many cases where your XML return may include two available engines and two available transmissions, but each transmission will be available with only one engine. This is good news, because it means that if you know which transmission is installed you also know which engine is installed, and vice-versa. It also allows you to pick precise MPG numbers, instead of using the range provided in the *fuel_efficiency_ratings* section for vehicles with multiple engines and/or transmission options.

```
<eng_trans_mpg_associations>
  <engine id="" standard="" installed="">
    <transmission id="" standard_for_engine="" installed="">
      <order_code></order_code>
      <invoice_price></invoice_price>
      <msrp_price></msrp_price>
      <fuel_efficiency>
        <mpg_city></mpg_city>
        <mpg_hwy></mpg_hwy>
      </fuel_efficiency>
    </transmission>
  </engine>
</eng_trans_mpg_associations>
```

Figure 4.8 – XML Syntax for *eng_trans_mpg_associations* element group

Breaking down the syntax from **Figure 4.8** line-by-line, we can see the section’s root tag, *eng_trans_mpg_associations*, contains a list of one or more *engine* elements, which have *id*, *standard*, and *installed* attributes. The engine *id* is unique to this engine, and can be used to reference detailed engine information from the *engines* section, described in section 4.4.3 of this document. The *standard* and *installed* attributes have the same meaning here as in the *engines* section. If the engine is standard for the vehicle style named in the *vehicle_style* parent tag, this will contain “Y.” If we can be certain that this engine is installed on this vehicle, *installed* will be set to “Y.”

All transmissions available to the engine identified by the *engine* element exist as child elements, as shown in the syntax example. Note that the same transmission may appear multiple times in the *eng_trans_mpg_associations* section if it is available with multiple engines. The *transmission* element has *id*, *standard_for_engine*, *installed* attributes. The *id* attribute references an id from the *transmissions* section described in section 4.4.4. The *standard_for_engine* attribute will be set to “Y” to indicate the default transmission for the engine identified in its parent element, and the *installed* attribute will be set to “Y” if it can be determined that the transmission is installed for the VIN provided.

The *transmission* element contains child elements *order_code*, *invoice_price*, *msrp_price*, and *fuel_efficiency*. These contain values that apply to this transmission when associated with the engine named in the parent element.

4.4.7 Features Section

The features section contains a significant amount of information about the equipment installed on the vehicle by default, at zero additional cost. Features are sometimes also referred to as standards or standard options. All features data is broken into categories. The *features* element will almost always contain more than one *category* child elements, and *category* elements often contain multiple *feature* child elements. All features have a category and a feature name. Many features have a value associated with them, while many only have a name and category.

```
<features>
  <category name="">
    <feature name="">
      <value></value>
    </feature>
  </category>
</features>
```

Figure 4.9 – XML Syntax for *features* element group

4.4.8 Options Section

The options section has a syntax similar to that of the features section. The *options* parent element generally contains multiple *category* child elements, which can contain one or more *option* child elements. All options have both a category and a name. Options that are always installed for the VIN number provided are given a value of “Y” for the *installed* attribute. The *option* element contains several child elements, which may or may not contain data, depending on the option. The *install_type* element indicates where the option is installed on the vehicle (at factory, port, or dealer).

```
<options>
  <category name="">
    <option name="" installed="">
      <install_type></install_type>
      <invoice_price></invoice_price>
      <msrp_price></msrp_price>
      <order_code></order_code>
      <description></description>
    </option>
  </category>
</options>
```

Figure 4.10 – XML Syntax for *options* element group

4.4.9 Colors section

The colors section contains data for exterior colors, interior colors, and where applicable, roof colors. RGB color codes are provided for all model year 2000 and later vehicles. The *is_two_tone* flag indicates whether a single RGB color applies, or if two RGB colors apply. Interior colors contain a *fabric_type* element, which at present time is unused.

```
<colors>
  <exterior_colors>
    <exterior_color mfr_code="">
      <basic_color_name></basic_color_name>
      <mfr_color_name></mfr_color_name>
      <primary_rgb_code r="" g="" b="" />
      <secondary_rgb_code r="" g="" b="" />
      <is_two_tone></is_two_tone>
    </exterior_color>
  </exterior_colors>
  <interior_colors>
    <interior_color mfr_code="">
      <basic_color_name></basic_color_name>
      <mfr_color_name></mfr_color_name>
      <primary_rgb_code r="" g="" b="" />
      <secondary_rgb_code r="" g="" b="" />
      <is_two_tone></is_two_tone>
    </interior_color>
  </interior_colors>
  <roof_colors>
    <roof_color mfr_code="">
      <basic_color_name></basic_color_name>
      <mfr_color_name></mfr_color_name>
      <primary_rgb_code r="" g="" b="" />
      <secondary_rgb_code r="" g="" b="" />
      <is_two_tone></is_two_tone>
      <fabric_type></fabric_type>
    </roof_color>
  </roof_colors>
</colors>
```

Figure 4.11 – XML Syntax for *colors* element group

4.4.10 Warranties Section

The warranties section can contain zero or more vehicle warranty records, for each type of warranty supplied with the vehicle. Please note that this warranty data reflects the warranties supplied with the vehicle when it was sold new, and are not necessarily still in effect for used vehicles.

```
<warranties>
  <warranty>
    <type></type>
    <name></name>
    <months></months>
    <miles></miles>
  </warranty>
</warranties>
```

Figure 4.12 – XML Syntax for *warranties* element group

4.4.11 Media Section

The media section contains links to single stock photos and lifestyle gallery images. Both full-size and thumbnail sizes are provided. Both single-stock and lifestyle gallery photos show a stock vehicle in a real-world driving environment. These images are provided for use in vehicle promotion only. Single stock photos are front-3/4 shots, while lifestyle gallery shots are given both a shot code and a shot name.

```
<media>
  <single_stock_photo>
    <stock_photo type="full" location="" />
    <stock_photo type="thumb" location="" />
  </single_stock_photo>
  <lifestyle_gallery>
    <lifestyle_photo shot_code="" shot_name="">
      <thumb_location></thumb_location>
      <full_location></full_location>
    </lifestyle_photo>
  </lifestyle_gallery>
</media>
```

Figure 4.14 – XML Syntax for *media* element group

4.4.12 Pricing Section

The pricing section contains prices for the base vehicle style when sold new without any optional engines, transmissions, or options included. This price does not reflect optional equipment that is installed by default for the VIN number provided. Prices for specific optional equipment are contained in the engines, transmissions, and options sections, and may be added to these prices to obtain a total cost for the vehicle, as sold new, with installed equipment.

```
<pricing>
  <msrp_price></msrp_price>
  <invoice_price></invoice_price>
  <destination_charge></destination_charge>
  <gas_guzzler_tax></gas_guzzler_tax>
</pricing>
```

Figure 4.13 – XML Syntax for *pricing* element group

4.4.13 Awards & Accolades Section (6.0.1 and later)

```
<awards_accolades>
  <award_accolade name="" source="">
    <website></website>
    <citation></citation>
    <snippet></snippet>
    <criteria>
      <engine id="" name="" />
      <transmission id="" name="" />
    </criteria>
  </award_accolade>
</awards_accolades>
```

Figure 4.15 – XML Syntax for *awards_accolades* element group

The awards and accolades section contains zero or more *award_accolade* elements, for each award associated with the vehicle. Each award record is given a name and source, either a website or a citation for offline media, and can contain a snippet of the award text. Award records always paint the vehicle in a positive light, and are intended to be used in the promotion of the vehicle. No negative reviews will be made available in this section.

Award records can be assigned engine and/or transmission-specific criteria, which indicate that the award applies to the vehicle only when the specified optional engine and/or transmission are installed.

4.4.14 eVox Mapping Section

The eVox mapping section provides a single eVox “VIF” identifier for the vehicle. For customers receiving eVox media either directly from eVox, or licensed through DataOne Software, this provides a method of selecting an appropriate set of eVox media for each decoded vehicle. At times, eVox does not make available a set of photos that exactly match the decoded vehicle. In these cases, a “best-fit” approach is taken, and the datapoints for which there is an exact match made between the decoded vehicle and the photographed vehicle are listed.

The *evox_mapping* element group will always contain zero or one *evox_match* elements. The *evox_match* element will always contain elements listing all datapoints used in comparison between the decoded and photographed vehicle. Each of these elements will have a value of ‘Y’ or ‘N’ to indicate an exact match has or has not been made.

```
<evox_mapping>
  <evox_match vif="">
    <matches_year></matches_year>
    <matches_make></matches_make>
    <matches_model></matches_model>
    <matches_trim></matches_trim>
    <matches_body_type></matches_body_type>
    <matches_cab_type></matches_cab_type>
    <matches_doors></matches_doors>
    <matches_drive_type></matches_drive_type>
  </evox_match>
</evox_mapping>
```

Figure 4.16 – XML Syntax for *evox_mapping* element group

5.0 Version update notes

Changes made between versions 6.0.0 and 6.0.1

- Attribute and element name changes in *eng_trans_epa_associations* element group.
- Removal of unused *fabric_type* element from interior colors section.
- Addition of *installed* attribute for *engine* and *transmission* elements.
- Addition of *Awards and Accolades* section (see 4.4.13)
- Addition of *eVox Mapping* section (see 4.4.14)
- Addition of *Fuel Efficiency* section (see 4.4.5)