TruSpeed Repair Connector API

Document Version 1.0

Table of Contents

1 Overview

MOTOR TruSpeed Repair is a web application for filtering, navigating, displaying and printing automotive information. TruSpeed Repair Connector allows for integration into complementary applications, bypassing end-user input to identify vehicle or part taxonomy for a specific query. This document covers the API for authentication and pop-up window/printing methods.

This API describes all of the parameters necessary for a customer's application to connect to TruSpeed Repair and initiate an end-user session.

2 TruSpeed Repair Connector API

The entry point to the TruSpeed Repair Connector is a single URL over HTTPS. By using a simple URL, customer applications are free to implement their web interface as they see fit; pop-up windows, frames and I-frames are all possible. To support this simplified model, the customer's application must instruct the end-user's web browser to navigate to this constructed URL.

Construction of the URL is two-fold. MOTOR provides a base HTTPS URL including some customer and product-specific parameters. The customer application constructs a query string of name-value pairs along with a SHA256 hash of the query string, current time, HTTP verb, and URI Path that is utilized with the private key. This hash is appended as a query string parameter in the URL.

To authenticate the customer call, the TruSpeed Repair Connector will create a hash based on all of the query string parameters excluding the parameter containing the hash. If the hashes match, and the embedded date-time stamp is within a specified valid time range, the end-user be allowed to enter the TruSpeed Repair web site.

2.1 Technology Requirements

The reader of this document should have experience with and access to the following technologies:

1. General understanding of SHA256 hashing, or similar
2. General understanding of HTTPS GET requests
3. A web development and test environment that supports browsing to the public internet over HTTPS

2.2 Glossary

Term	Definition
API	Application Programming Interface. An API is a set of rules and specifications that software programs can follow to communicate with each other.
Authentication	Establish the authenticity of or prove genuine.
Authorization	Granting or denying access to a resource based on principals and credentials. This specification does not support any level of authorization other than full access.
Hash	An algorithm that creates an effectively unique message given any input data. A hash can be used to generate a "signature" of the input data.
PIN	Personal Identification Number. Text string used to uniquely identify an end user.
SHA256	Secure Hash Algorithm 256-bit. SHA256 is a hashing algorithm available in most programming languages and frameworks.
Signature	A unique message generated by processing the content of a message. See Hash.
UNIX Time	A way to represent a timestamp by the number of seconds since January 1st, 1970, at 00:00:00 UTC
URL	Uniform Resource Locator. Most commonly used to refer to a web site address.

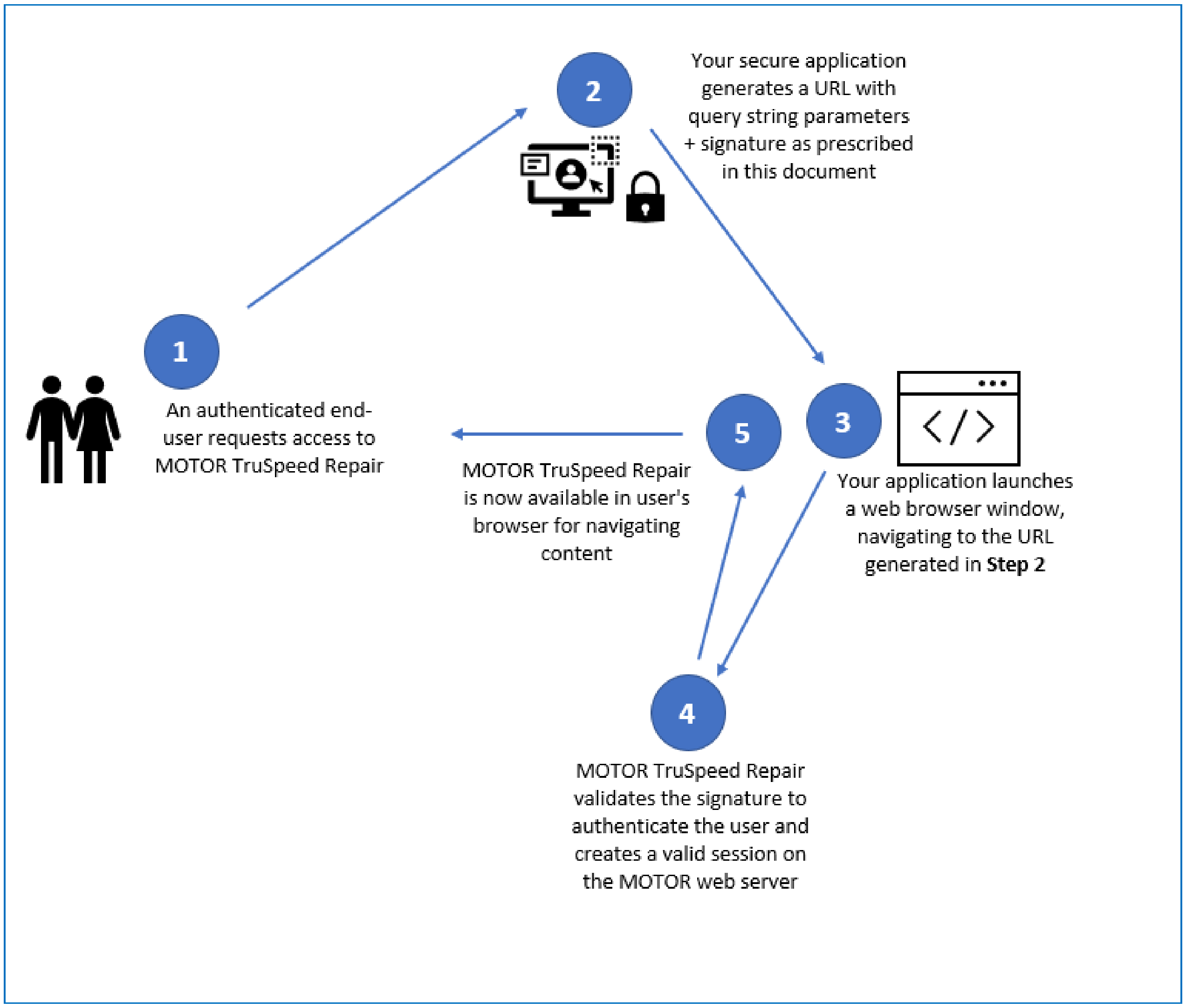
3 Authentication

3.1 Approach

To keep the process as simple *and* robust as possible, MOTOR has taken the approach of utilizing a simple and effective hash of the incoming data over HTTPS in addition to date/time stamps to authenticate incoming requests.

3.2 HTTPS Request / Response Flow

This diagram, detailed in the steps section below, indicates the architectural view of the process.


2
Your secure application generates a URL with query string parameters + signature as prescribed in this document
1
An authenticated end-user requests access to MOTOR TruSpeed Repair
5
MOTOR TruSpeed Repair is now available in user's browser for navigating content
3
Your application launches a web browser window, navigating to the URL generated in **Step 2**
4
MOTOR TruSpeed Repair validates the signature to authenticate the user and creates a valid session on the MOTOR web server

Processing Flow Diagram

1. An authenticated end-user requests access to MOTOR-hosted content
    a. Note: Here, authenticated means a valid/confirmed/authenticated end-user of the customer's application.
2. Customer's application then generates a URL as prescribed in this document. Example URL would look like:

```
https://sites.motor.com/connector?pin=8675309&vin=77777777X7XXXXXXX&year=2009&ma
ke=Chevrolet&model=Cobalt&Scheme=Shared&XDate=1658288073&ApiKey=VrYH0mG0ED&Sig=Z
S8OjEmL%2BOi22Dpmz8VIrjFW%2BAwh4jt2MfcN6bVclDE%3D
```

3. Customer's application launches a web browser instance with the URL generated in Step 2.
4. TruSpeed Repair Connector builds a signature hash from the query string parameters and compares it to the customer provided signature to authenticate the end-user request. If the signatures do not match, an "Authentication Error" message will be displayed.
5. MOTOR TruSpeed Repair is now available for interaction.

Upon success of the above flow, the end-user is then presented with the customized TruSpeed Repair web site.

3.2.1 Application Code Flow

The construction of the URL sent to MOTOR follows these steps:

1. Construct a valid query string containing name-value pairs for each field.
2. Construct the signature with the query string and private key
3. Build the HTTPS URL for requesting access to the MOTOR-hosted website.

3.2.2 Building the URL to be Posted

For ease of implementation, there are no web-service calls or other glue code required. The URL is simply sent over HTTPS to the TruSpeed Repair Connector production site at https://sites.motor.com/connector via an HTTPS/GET request. Here is an example:

```
https://sites.motor.com/connector?pin=8675309&year=2009&make=Chevrolet&model=Cob
alt&Scheme=Shared&XDate=1658324390&ApiKey=VrYH0mG0EC&Sig=iWqKfpLIPlQI8K5%2FZ5S88
qdLPOk7C8cn4AyuGVn9rhk%3D
```

Note that this sample URL will not work. The XDate must be valid and the ApiKey must be valid.

3.3 Public and Private API Keys

To interact with the application, public and private keys are required for authentication. Every public key is mapped to only one private key. The public key is similar to a public identity like a username. The private key is similar to a password and is used to sign requests. The private key is never sent when making any web requests. API keys should be kept private and never distributed.

3.4 Signature Construction

The signature is built using a one-way hashing algorithm and a private key. The signature algorithm works as follows:

1. Take the constructed query string parameters of ApiKey (Public Key)
2. Calculate the SHA256 hash value for the concatenated string of query parameters, HTTP verb - GET, POST, PUT, DELETE, time stamp - converted to UNIX epoch, and URI Path, that are new line separated with the private key
3. The hash value is Base64 encoded which converts binary data to an ASCII string
4. The hash value is URL Encoded which replaces unsafe ASCII characters with a "%" and replaces a space with a plus (+) sign or with %20
5. The hash value is then appended to the query string with the parameter name of signature:
    a. For example: &sig=nQppVAKc21Ui12W2lwJb%2FtKOX5koS%2FAKQsJNTEwpIqA%3D

Here is a sample code of how signature is constructed.

```
function getSignature() {
    const publicKey = document.querySelector('#publicKey').value;
    const privateKey = document.querySelector('#privateKey').value;

    const epoch = Math.floor(Date.now() / 1000);
    const pathname = '/connector';
    const toSign = publicKey + '\nGET\n' + epoch + '\n' + pathname;
    const hash = CryptoJS.HmacSHA256(toSign, privateKey);
    let signature = CryptoJS.enc.Base64.stringify(hash);
    signature = encodeURIComponent(signature);
    return signature;
}
```

3.4.1 Validation Code

Upon receipt of the HTTPS/GET, the request will be validated by the MOTOR TruSpeed Repair solution by performing the following steps:

1. Calculate the signature (hash) on query string sans the signature parameter (treating the input text as UTF-8)
2. Ensure that the newly calculated signature and the supplied signature are the same.
3. Ensure the time stamp that is used in a request must exactly match the time stamp used in generating the authentication signature
    a. Note that the time stamp must fall within 15 minutes plus or minus of the server time.
    b. Additionally, even though the MOTOR-hosted web site will be in EST (Eastern Standard Time), it is required that this time be UTC.

4 Parameters

4.1 Customer Name-Value Parameters for Content

Gather the necessary name-value pairs needed for authentication and execution of the application. All required URL parameters must be provided on every call.

Field / Description	Query String Parameter Name	Value	Required?	Sample Query String Parameter
PIN	pin	Any value up to 256 characters can be used. Common values include a customer ID or the end user's email address.	Required	pin=8675309
Current Date/Timestamp This time stamp is used to prevent any end-user from re-requesting access to the application by simply book-marking the web site.	XDate	The current date-time expressed in UTC time. The format of the date-time stamp should be in UNIX Epoch Time Stamp. *Due to the difficulties of synchronizing clocks there will be an allowance for mis-synchronization.*	Required	XDate=1658285514
Api Key	ApiKey	The Public Key (Case Sensitive)	Required	ApiKey=VrYi0mG0EC

Field / Description	Query String Parameter Name	Value	Required?	Sample Query String Parameter
Scheme	Scheme	Scheme represents the type of authentication being used The Sample Query String Parameter "Shared" validates that public key and private key is being used for authentication	Required	Scheme=Shared
Signature	Sig	Generated Hash value used for authentication	Required	Sig=ZS8OjEmL%2BOi22Dpmz8VIrjFW%2BAwh4jt2MfcN6bVclDE%3D
VehicleToEngine ConfigId VehicleToEngine ConfigId is component of the Vehicle Configuration Database (VCdb). Refer to section 4.2 for more information about Vehicle Attributes.	vehicleToEngineConfigID	Unique identifier which fully resolves to a vehicle based on the Year, Make, Model, Submodel, and Engine	Optional Refer to section 4.3 for "Vehicle Parameter Precedence"	vehicleToEngineConfigId=207
VIN	Vin	Vehicle Identification Number (VIN) Note, 10 characters are required for VIN. Any excess characters are ignored.	Optional Refer to section 4.3 for "Vehicle Parameter Precedence"	vin=0123456789XXXXXXX

Field / Description	Query String Parameter Name	Value	Required?	Sample Query String Parameter
Year The model year of the vehicle in question	Year	The vehicle year	Optional Refer to section 4.3 for "Vehicle Parameter Precedence"	year=2010
Make The make of the vehicle in question. Refer to section 5.1 to view all supported Make values	Make	The full name of the make	Optional Refer to section 4.3 for "Vehicle Parameter Precedence"	make=Chevrolet
Model The model of the vehicle in question	Model	The full name of the model	Optional	model=tahoe
Search Term	searchTerm	Textual search string that will drive the search results displayed	Optional	searchTerm=bumper

Construct a valid HTTPS query string containing a name-value pair for each parameter to be passed. The first parameter's name is prefixed with a question mark (?) and each subsequent is prefixed with an ampersand (&). Following each parameter name is an equal sign (=) followed by its associated value.

MOTOR will use the order of the parameters in the URL when calculating the hash value.

A valid query string (excluding the signature) using some of the contents of the "Sample Query String Parameter" above would be:

```
?pin=8675309&vin=77777777X7XXXXXXX&year=2009&make=Chevrolet&model=Cobalt&searchT
erm=front bumper&Scheme=Shared&XDate=1658290023&ApiKey=VrYH0mG0EC
```

4.2 Vehicle Attributes (VCdb)

Vehicle attributes are extended parameters that can be utilized in TruSpeed Repair. All vehicle attributes are derived values from the VCdb (Vehicle Configuration database), an Auto Care Association Data Standard. To translate these vehicle attributes, you must have a valid subscription to the Auto Care Association VCdb standard. For more information, refer to https://www.autocare.org/.

Vehicle attributes are ID fields that make up the vehicle, equipment, system or engine configuration which is used by TruSpeed Repair to further filter a vehicle. The ID fields are references to attribute values available in VCdb to further distinguish/filter a vehicle. For example, ID 1 for Fuel Type ID represents a "Gas" vehicle.

Vehicle attributes are not required to be provided to browse content in the application. BaseVehicleID is an optional paramater if no other extended vehicle attributes are provided. However, for vehicle attributes to be evaluated BaseVehicleID is then required. One or more vehicle attributes can be utilized with the inclusion of BaseVehicleID. If more than one vehicle attributes is utilized, with the inclusion of BaseVehicleID, it will use the "AND" logic and those attributes will be combined to filter down to a vehicle.

The following table defines all supported Vehicle Attributes.

Field / Description	**Query String Parameter Name**	**Value**	**Required?**	**Sample Query String Parameter**
BaseVehicleID	baseVehicleID	Represents the Year, Make, and Model of a Vehicle	Optional Note, BaseVehicleID is required if any of the following fields in this table are utilized	baseVehicleId=119814
SubModelID	subModelID	Represents the SubModelNam e of a Vehicle	Optional Note, BaseVehicleID is required if this field is utilized	subModelId=1591
EngineDesignationID	engineDesignation ID	Represents the Engine Designation Name from the OEM	Optional Note, BaseVehicleID is required if this field is utilized	engineDesignationId=1
EngineVINID	engineVINID	Represents the VIN the OEM designates to represent the Engine	Optional Note, BaseVehicleID is required if this field is utilized	engineVINId=19
EngineVersionID	engineVersionID	Represents a reference to the family of engines as designated by the manufacturer	Optional Note, BaseVehicleID is required if this field is utilized	engineVersionId=3

Field / Description	**Query String Parameter Name**	**Value**	**Required?**	**Sample Query String Parameter**
EngineMfrID	engineMfrID	Represents the manufacturer of the engine	Optional Note, BaseVehicleID is required if this field is utilized	engineMfrId=22
PowerOutputID Power Output only exists for European Makes between the years 1985 and 2009. For all Light Duty applications, it exists for years 2010 and later	powerOutputID	Represents the Power Output as specified by the OEM manufacturer that is calculated in Horsepower (HP) and then converted into Kilowatts (Kw)	Optional Note, BaseVehicleID is required if this field is utilized	powerOutputId=363
ValvesID	valvesID	Represents the number of intake and exhaust valves per engine	Optional Note, BaseVehicleID is required if this field is utilized	valvesId=7
FuelDeliveryTypeID	fuelDeliveryTypeID	Represents only of two primary types – carbureted or fuel injected	Optional Note, BaseVehicleID is required if this field is utilized	fuelDeliveryTypeId=5
FuelDeliverySubTypeID	fuelDeliverySubTypeID	Represents a secondary segmentation of the Fuel Delivery System	Optional Note, BaseVehicleID is required if this field is utilized	fuelDeliverySubTypeId=5
FuelSystemControlTypeID	fuelSystemControlTypeID	Represents two values of fuel system, if electronically or mechanically controlled	Optional Note, BaseVehicleID is required if this field is utilized	fuelSystemControlTypeId=5
FuelSystemDesignID	fuelSystemDesignID	Represents an extension of Fuel Delivery System that often represents the original manufacturer	Optional Note, BaseVehicleID is required if this field is utilized	fuelSystemDesignId=6

Field / Description	Query String Parameter Name	Value	Required?	Sample Query String Parameter
FuelTypeID	fuelTypeID	Represents the fuel type as specified by the OEM	Optional Note, BaseVehicleID is required if this field is utilized	fuelTypeId=7
AspirationID	aspirationID	Represents the air intake system	Optional Note, BaseVehicleID is required if this field is utilized	aspirationId=5
CylinderHeadTypeID	cylinderHeadTypeID	Represents the valve train configuration as designated by the vehicle manufacturer	Optional Note, BaseVehicleID is required if this field is utilized	cylinderHeadTypeId=6
IgnitionSystemTypeID	ignitionSystemTypeID	Represents the technology of the primary ignition system	Optional Note, BaseVehicleID is required if this field is utilized	ignitionSystemTypeId=6

4.3 Vehicle Parameters Precedence

There are multiple mechanisms to define a vehicle. These items are evaluated in a priority order. A higher priority item is always evaluated first and supersedes any lower priority item. If an invalid value is provided for a parameter, it will discard that parameter and evaluate the vehicle based on the next highest priority item that has a valid value. The vehicle defining parameters are evaluated in the following descending order:

1. VehicleToEngineConfigId
2. BaseVehicleID + Engine Parameters (e.g., EngineMfrID, EngineVINID, EngineVersionID)
3. VIN
4. Year + Make

The only exception occurs when evaluating BaseVehicleID + supplied Engine Parameters. If, for example, three parameters are supplied, but one is invalid; then the invalid item is discarded and the remaining two (plus the BaseVehicleID) are evaluated to determine the Vehicle (assuming other, higher priority, parameters are not supplied).

The following table highlights some scenarios of vehicle parameters and how a vehicle is evaluated according to the defined precedence.

Parameters Used	Result	Explanation
• *VALID* – VehicleToEngineConfigId • *VALID* – VIN • *VALID* – Year + Make	Vehicle is defined based on VehicleToEngineConfigId	Since VehicleToEngineConfigId has the highest priority, the vehicle will be defined based on VehicleToEngineConfigId.
• ***INVALID*** – VehicleToEngineConfigId • *VALID* – VIN • *VALID* – Year + Make	Vehicle is defined based on VIN	Though VehicleToEngineConfigId has the highest priority, with the list of parameters used, an invalid value was provided. The vehicle will be defined based on the next highest priority which is VIN.
• ***INVALID*** – VIN • *VALID* – Year + Make	Vehicle is defined based on Year + Make	Though VIN has the highest priority, with the list of parameters used, an invalid value was provided. The vehicle will be defined based on the next highest priority which is Year + Make.
• ***INVALID*** – VehicleToEngineConfigId • ***INVALID*** – BaseVehicleID + ***INVALID*** – Engine Parameters • *VALID* – Year + Make	Vehicle is defined based on Year + Make	Though VehicleToEngineConfigId and BaseVehicleID + Engine Attributes have a higher priority, with the list of parameters used, invalid values were provided. The vehicle will be defined based on the next highest priority which is Year + Make.
• *VALID* – BaseVehicleID + *VALID*/**I*NVALID*** – Engine Parameters • ***INVALID*** – VehicleToEngineConfigId • *VALID* – Year + Make In this scenario, note that for Engine Parameters, one valid parameter and one invalid parameter were used.	Vehicle is defined based on BaseVehicleID and Valid Engine Parameters	Though VehicleToEngineConfigId has the highest priority, with the list of parameters used, an invalid value was provided. The vehicle will be defined based on the next highest priority which is BaseVehicleID + Valid Engine Parameter. It will discard the invalid Engine Parameter.

5 Supplemental Information

5.1 Supported "Make" Values

This section highlights the supported string values available in VCdb for the parameter "Make".

- Acura
- Alfa Romeo
- American Motors
- Audi
- BMW
- Buick
- Cadillac
- Chevrolet
- Chrysler
- Daewoo
- Daihatsu
- Dodge
- Eagle
- Fiat
- Ford
- Genesis
- Geo
- GMC
- Honda
- Hummer
- Hyundai
- INFINITI
- Isuzu
- Jaguar
- Jeep
- Kia
- Land Rover
- Lancia
- Lexus
- Lincoln
- Mazda
- Mercedes-Benz
- Mercury
- Merkur
- Mini
- Mitsubishi
- Nissan
- Oldsmobile
- Opel
- Peugeot
- Pontiac
- Porsche
- Ram
- Renault
- Saab
- Saturn
- Scion
- Smart
- Sterling
- Subaru
- Suzuki
- Toyota
- Volkswagen
- Volvo
- Yugo

Document History

Version	Date	Author	Comments
1.0	11/9/2022	MOTOR Information Systems	Initial Release